UNIVERSIDADE FEDERAL DE MINAS GERAIS
Faculdade de Letras
Programa de Pós-Graduação em Estudos Linguísticos

José Carlos da Costa Júnior

Padrão informacional de stanzas de pacientes com esquizofrenia

BELO HORIZONTE
2022

José Carlos da Costa Júnior

Padrão informacional de stanzas de pacientes com esquizofrenia

Tese de doutorado apresentada ao Programa de Pós-Graduação em Linguística da Universidade Federal de Minas Gerais como requisito parcial para obtenção do título de Doutor em Estudos Linguísticos

Orientadora: Profa. Dra. Heliana Mello
Coorientador: Prof. Dr. Bruno Rocha

Belo Horizonte
2022

Costa Júnior, José Carlos da.
C837p Padrão informacional de stanzas de pessoas com esquizofrenia [manuscrito] / José Carlos da Costa Júnior. – 2022.
1 recurso online (251 f. : il., tabs., color., p&b.) : pdf.

Orientadora: Heliana Ribeiro de Mello.

Coorientador: Bruno Rati de Melo Rocha.

Área de concentração: Estudos Linguísticos Baseados em Corpora.

Linha de Pesquisa: Linguística Teórica e Descritiva.

Tese (doutorado) – Universidade Federal de Minas Gerais, Faculdade de Letras.

Bibliografia: f. 132-135.

Anexos: f. 136-251

Exigências do sistema: Adobe Acrobat Reader.

1. Atos de fala (Linguística) – Teses. 2. Esquizofrenia – Teses. 3. Linguística – Teses. I. Mello, Heliana. II. Rocha, Bruno Rati de Melo. III. Universidade Federal de Minas Gerais. Faculdade de Letras. IV. Título.

CDD: 410

Ficha catalográfica elaborada pelo Bibliotecário Israel José da Silva – CRB/6-2128

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FACULDADE DE LETRAS

PROGRAMA DE PÓS-GRADUAÇÃO EM ESTUDOS LINGUÍSTICOS

FOLHA DE APROVAÇÃO

Padrão informacional de stanzas de pacientes com esquizofrenia

JOSÉ CARLOS DA COSTA JUNIOR

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ESTUDOS LINGUÍSTICOS, como requisito para obtenção do grau de Doutor em ESTUDOS LINGUÍSTICOS, área de concentração LINGUÍSTICA TEÓRICA E DESCRITIVA, linha de pesquisa Estudos Linguísticos Baseados em Corpora.
Aprovada em 25 de fevereiro de 2022, pela banca constituída pelos membros:

Prof(a). Heliana Ribeiro de Mello - Orientadora
UFMG

Prof(a). Bruno Neves Rati de Melo Rocha - Coorientador
UFMG

Prof(a). Tommaso Raso
UFMG

Prof(a). João Vinicius Salgado
UFMG

Prof(a). Waldemar Ferreira Netto
USP

Prof(a). Natália Bezerra Mota
UFRJ

Belo Horizonte, 25 de fevereiro de 2022.

Documento assinado eletronicamente por Tommaso Raso, Professor do Magistério Superior, em 25/02/2022, às 19:08, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por Heliana Ribeiro de Mello, Professora do Magistério Superior, em 25/02/2022, às 19:13, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por João Vinicius Salgado, Membro de comissão, em 02/03/2022, às 16:46, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por Waldemar Ferreira Netto, Usuário Externo, em 04/03/2022, às 15:54, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por Natália Bezerra Mota, Usuária Externa, em 06/03/2022, às 08:24, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por Bruno Neves Rati de Melo Rocha, Professor do Magistério Superior, em 17/03/2022, às 15:21, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador 1224592 e o código CRC 6ED0896F.

A Erwann Entem

AGRADECIMENTOS

Agradeço a meu marido Erwann Entem pela enorme paciência, pelas mensagens na porta da geladeira das quais não se pode esquecer e pela proximidade em momentos que pareciam não ter fim.

Também agradeço aos professores e colaboradores do LEEL que, de uma forma ou de outra, contribuíram para que este trabalho existisse.

Ademais, agradeço ao CNPq por ter financiado minha pesquisa.

Resumo

Objetivos: Fazer uma comparação entre o padrão informacional e disfluências da fala de pacientes com esquizofrenia e participantes sem essa condição. Esses padrões representam os possíveis arranjos de unidades informacionais encontrados em enunciados (CRESTI, 2000; RASO, 2012; MONEGLIA, RASO, 2014). Além disso, é proposta uma descrição preliminar das medidas da fala, léxico e classes de palavras desses pacientes. Hipóteses: I) indivíduos com esquizofrenia possuiriam padrão informacional menos variado e complexo – medido pela quantidade de unidades textuais do enunciado - do que indivíduos sem esquizofrenia, tal como foi sugerido em Cresti et al (2015) II) esses pacientes realizariam mais disfluências, como pausas preenchidas e silenciosas com maior duração, maior quantidade de reformulações e enunciados interrompidos do que os participantes sem esquizofrenia. Metodologia: Foram utilizados dois corpora orais representativos de suas populações, C-ORAL-ESQ (FERRARI; ROCHA; RASO, em construção), e minicorpus de monólogos do C-ORAL-BRASIL (RASO; MELLO, 2012), a partir dos quais foram selecionados seis textos etiquetados informacionalmente em cada corpus. O parâmetro de comparação entre os grupos é a quantidade de Comentários Ligados – COBs – unidade informacional textual cuja presença categoriza uma unidade terminada da fala como stanza. Dessa forma, são comparadas stanzas com igual quantidade de COBs e contabilizadas as outras unidades textuais, uma vez que a presença dessas indicaria a complexidade maior ou menor de elaboração da fala. Para isso, foram desenvolvidos programas em linguagem Python (VAN ROSSUM, DRAKE, 1995), disponibilizados para outros usuários e pesquisas, que extraem o texto das transcrições, contam todas as etiquetas e disfluências, transformam português semiortográfico em padrão, etiquetam com classes gramaticais detalhando sua morfologia e frequência; fazem análise estatística, criam corpus de pesquisa balanceado e geram visualizações de todas as variáveis comparadas entre os grupos automaticamente, entre outras implementações. Isso foi realizado principalmente com o auxílio de bibliotecas como o Pandas (MCKINNEY, 2010), para análise estatística e automação, Natural Language Toolkit (BIRD et al, 2011) e *re* (VAN ROSSUM, 2020) para processamento de linguagem natural e mineração de texto. Resultados: Pacientes com esquizofrenia realizaram menos: padrões únicos, unidades textuais totais ($p = 0,001$); Tópicos ($p = 0,01$); Parentéticos ($p = 0,03$), Introdutores Locutivos

(p =0,01), Comentários Múltiplos (p = 0,04) entre outros resultados estatisticamente significativos no teste U de Mann Whitney, em diferentes amostras. Enunciados interrompidos foram mais frequentes no C-ORAL-BRASIL, mas foi constatado que os pacientes realizaram menos unidades textuais antes de sua interrupção. Retractings, por sua vez, tiveram registro variado nas amostras, mas foram mais frequentes no C-ORAL-BRASIL, com similar distribuição de sua dimensão silábica e segmento inicial. Quanto às pausas, foi verificado que os participantes do C-ORAL-ESQ as realiza com maior duração tanto para as silenciosas (p < 0,01) quanto para as preenchidas, embora esta última não tenha significância estatística. Portanto, é defendido que pacientes com esquizofrenia possuem fala menos elaborada do ponto de vista informacional – e consequentemente menos variada melodicamente -, além de terem mais disfluências nas variáveis medidas de uma forma geral.

Palavras-chave: Esquizofrenia. Padrão informacional. Teoria da Língua em Ato. Comparação entre corpora orais. Processamento de linguagem natural – PLN.

Abstract

Goals: We make a comparison between the informational pattern and speech disfluencies of patients with schizophrenia and participants without this mental illness. The informational pattern represents the different arrays of information units that are found in utterances (CRESTI, 2000; RASO, 2012; MONEGLIA, RASO, 2014). We also offer a preliminary description for speech disfluencies, lexicon and part of speech of a representative sample of patients. Hypotheses: I) patients with schizophrenia could have a less varied and less complex informational pattern - given for the textual units frequency- as it was already suggested in Cresti et al (2015); II) these patients could make more speech disfluencies, as larger filled and silent pauses or more speech reformulations or interrupted utterances than people without schizophrenia. Methodology: We use 12 informational tagged transcriptions from two oral corpora - 6 of each -, namely C-ORAL-ESQ (FERRARI, ROCHA, RASO, forthcoming), which is representative of patients with schizophrenia, and C-ORAL-BRASIL minicorpus (RASO, MELLO, 2012), which is representative of spoken Brazilian Portuguese. Comparison between the two groups was based on the number of Bound Comments - COBs - a textual unit whose presence defines a terminated unit of speech as a stanza. We compare stanzas with the same number of COBs and we count other textual units around, as these last ones could suggest more complexity in speech elaboration. For this purpose, it was developed Python (VAN ROSSUM, DRAKE, 1995) applications, which are free available for users, that extract text transcriptions and normalize their spelling conventions, tag part of speech, detail word frequency and morphology, make statistical analysis, build a balanced research sample and generate charts of all variables compared automatically, among other implementations. This was built mainly with Pandas (MCKINNEY, 2010), for statistical analysis and automation; Natural Language Toolkit (BIRD et al, 2011) and buil-in libraries like re (VAN ROSSUM, 2020) for natural language processing and text mining. Results: Patients with schizophrenia made less: unique patterns, total textual units ($p = 0{,}001$); Topics ($p=0{,}01$); Parenthetics ($p=0{,}03$); Locutive Introducers ($p=0{,}01$); Multiple Comments ($p=0{,}04$) among other significant results in Mann Whitney U test. Interrupted utterances were more frequent in C-ORAL-BRASIL, although patients with schizophrenia made less textual units before their interruption. Retractings, for their turn, presented different

frequency patterns in both groups, though they were less frequent in patient's speech in the most relevant samples, with similar number of syllables and distribution of the first phonetic segment. Regarding pauses, it was observed that patients made them longer than no patients, both for silent ($p<0,01$) as for filled pauses, though these last ones had no statistical significance. Therefore, we claim that patients with schizophrenia have less elaborated speech from the point of view of the information pattern – hence less varied melodically - as well as they have more speech disfluencies in the considered variables overall.

Key-words: Schizophrenia. Informational pattern. Language into Act Theory. Oral corpora comparison. Natural language processing – NLP.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

+: Enunciados interrompidos

/: Quebra prosódica não terminal

//: Quebra prosódica terminal

<>: Começo e fim de fala sobreposta

ADJ: Adjetivos

ADV: Advérbios

ALL: Alocutivo

APC: Apêndice de Comentário

APT: Apêndice de Tópico

ART: Artigos

AUX: Unidades dialógicas

CMM: Comentários Múltiplos

CNT: Conativo

COB: Comentário Ligado

COM: Comentário

DCT: Conector discursivo

DELAF: Dicionário de Palavras Simples Flexionadas para o Português Brasileiro

EMP: Unidade vazia

EXP: Expressivo

IN: Interjeições

INP: Incipitário

INT: Introdutor Locutivo

KC: Conjunções coordenadas

KS: Conjunções subordinadas

LEEL: Laboratório de Estudos Empíricos e Experimentais da Linguagem

N: Substantivos

NILC: Núcleo Interinstitucional de Linguística Computacional

NLTK: Natural Language Toolkit

NPROP: Nomes próprios

NUM: Numerais

PANSS: Escala das Síndromes Positiva e Negativa

PAR_PRL: Parentético ou lista de Parentéticos

PAUSA_P ou TMT: Pausa preenchida

PCP: Particípios passado

PHA: Fático

PREP|+: Preposições polilexicais

PRO-KS: Pronomes conectivo subordinativo

PRO-KS-REL: Pronomes conectivo subordinativo

PROADJ: Pronomes adjetivos

PROPESS: Pronomes pessoais

PROSUB: Pronomes substantivos

SCA: Escansão

SN: Sintagma nominal

SP: Sintagma preposicional

SV: Sintagma verbal

TAG_r: Unidades reportadas

TOP_TPL: Tópico ou lista de Tópico

UNC: Unidade que não foi possível ser reconhecida

V: Verbos

VAUX: Verbos auxiliares

[/d] ou /d - no qual d representa qualquer número seguido de barra: Retractings

africadas_des: Africadas desvozeadas

africadas_voz: Africadas vozeadas

fricativas_des: Fricativas desvozeadas

oclus_des: Oclusivas desvozeadas

oclus_nas: Oclusivas nasais

oclus_voz: Oclusivas vozeadas

Sumário

1 INTRODUÇÃO E OBJETIVOS

Norteado principalmente pela Teoria da Língua em Ato (CRESTI, 2000; RASO, 2012; MONEGLIA, RASO, 2014), este trabalho tem o objetivo principal de descrever e comparar o padrão informacional de pacientes com esquizofrenia e pessoas sem essa condição. O padrão informacional diz respeito aos distintos arranjos de unidades informacionais presentes em enunciados e pode revelar o quão elaborada é a fala desses pacientes.

Além disso, outro objetivo desta tese é propor uma descrição preliminar do léxico – como frequência de classes de palavras e detalhes morfológicos - e de medidas da fala – como duração e quantidade de enunciados, pausas e retractings - de uma amostra representativa do C-ORAL-ESQ (FERRARI, ROCHA, RASO, em construção).

Tanto a comparação em questão quanto a descrição foram apoiados por diversos programas desenvolvidos em linguagem Python[1] pelo autor desta tese e disponibilizados para usuários que queiram estudar os fenômenos linguísticos aqui discutidos, quer no C-ORAL-ESQ, quer no C-ORAL-BRASIL (RASO, MELLO, 2012). Dessa forma, é possível trabalhar com esses dois corpora e realizar procedimentos complexos, discutidos ao longo deste trabalho, na maioria das vezes apenas com uma conta do gmail. Isso é feito pelo Google Colab, que é um ambiente virtual para codificação e execução de programas principalmente em linguagem Python, sem necessidade de instalação no computador do usuário. Como a grande maioria desses códigos foi amplamente automatizada, não são necessários conhecimentos de programação para utilizá-los. Consequentemente, o usuário tem sua pesquisa facilitada nos corpora mencionados – que fornecem as transcrições - e a possibilidade de estudar fenômenos linguísticos de pacientes com esquizofrenia, tal como discutido neste trabalho.

Nessas diversas codificações, foi feito um esforço para integrar estudos valiosos do português brasileiro e disponíveis em Python, como o Mac-Morpho (ALUÍSIO et al, 2003), além de dicionários como o DELAF- Dicionário de Palavras Simples Flexionadas para o Português Brasileiro (MUNIZ, 2003; 2004) e a VerboWeb

[1]Todos os códigos estão disponíveis publicamente em meu perfil do Github: https://github.com/carlosjuniorcosta1

(CANÇADO et al, 2017) para enriquecer as possibilidades de descrição do léxico dos pacientes - também aplicáveis ao C-ORAL-BRASIL. Isso foi realizado porque também foi possível desenvolver a normalização automática do português semiortográfico das transcrições do C-ORAL-ESQ para português padrão. Assim, fenômenos importantes da fala como aférese (*bacaxi, bacate*) ou gramaticalização de verbos (*xá* eu fazer no lugar de *deixa* eu fazer, por exemplo) são preservados, e a etiquetagem de classes de palavras, a lematização e as informações morfológicas ou sintáticas, que requisitam português padrão, por exemplo, são viabilizadas com boa qualidade e de forma gratuita para os usuários que queiram estudar futuramente pessoas com esquizofrenia utilizando o C-ORAL-ESQ.

1.1 Justificativa

Este trabalho se justifica porque oferece a possibilidade e a tecnologia para descrever quantitativamente e de forma estatística como a fala dos pacientes é organizada do ponto de vista informacional. Entender a complexidade dos padrões informacionais, seja com a frequência de determinadas unidades, seja com sua ausência, pode ajudar, futuramente, a auxiliar no diagnóstico da esquizofrenia. A contagem e classificação das diversas disfluências analisadas nesta tese também podem ajudar nesse sentido, uma vez que maior ou menor quantidade delas, tais como duração e frequência de pausas preenchidas, quantidade de reformulações e enunciados interrompidos, poderiam fornecer pistas importantes sobre a fala dos pacientes analisados quando comparados a pessoas sem esquizofrenia.

Além disso, as ferramentas computacionais disponibilizadas para o tratamento do léxico permitem que outros estudiosos comparem alguns fenômenos que, neste trabalho, são trazidos apenas em forma de descrição, já que apenas os padrões informacionais e medidas da fala são comparados com pessoas que não têm esquizofrenia, e não o léxico.

1.2 Organização do texto

Este trabalho se organiza conforme a seguir. No capítulo II, são apresentados os pressupostos teóricos basilares para esta tese, com foco na Teoria da Língua em Ato e em estudos que comparam fenômenos linguísticos em pacientes e não

pacientes. No capítulo III, é explicitada a metodologia utilizada para a comparação entre o C-ORAL-ESQ e o C-ORAL-BRASIL, assim como os procedimentos utilizados para a descrição de uma amostra representativa do C-ORAL-ESQ. No capítulo IV, é primeiramente apresentada uma descrição preliminar de medidas da fala e disfluências de uma amostra representativa, com 16 áudios, do C-ORAL-ESQ. Posteriomente, ainda no capítulo IV, são apresentadas as palavras e classes gramaticais mais frequentes, além de detalhes fonológicos e fonéticos das classes lexicais de maior ocorrência na fala dos pacientes. No capítulo V, é realizada a comparação entre o padrão informacional e medidas da fala entre pacientes e não pacientes, às quais seguem as considerações finais desta tese. Por fim, todos os programas aqui utilizados são apresentados nos Anexos, assim como instruções de uso.

2 PRESSUPOSTOS TEÓRICOS

Este capítulo contempla a fundamentação teórica utilizada neste trabalho. Na seção 2.1, é apresentada a definição e principais sintomas da esquizofrenia, além de um panorama de problemas linguísticos já estudados tendo por base pacientes com essa condição. Na seção 2.2, são apresentados os principais pressupostos da Teoria da Língua em Ato (RASO, 2012; MONEGLIA, RASO, 2014). Já das seções 2.3 a 2.4 são discutidos alguns trabalhos relevantes que comparam pacientes e não pacientes. Na seção 2.3, é apresentada uma pesquisa de Cresti et al (2015), que discute o padrão informacional de pacientes e não pacientes em um corpus oral do italiano. Na seção 2.4 é discutido o trabalho de Steuber (2011), que compara diversas disfluências de pessoas com esquizofrenia e com indivíduos com depressão. Posteriormente, são trazidos à tona diversos trabalhos que comparam pausas – preenchidas ou não – na fala desses pacientes na seção 2.5. A seguir, na seção 2.6, é apresentado um estudo de Mota et al (2017), que utiliza computação para gerar grafos que representam a possível desorganização da fala de pessoas com esquizofrenia, comparando-as com indivíduos com transtorno bipolar e um grupo controle. Por fim, na seção 2.7, é apresentado o estudo de Jorge (2020), a qual compara parâmetros prosódicos de pessoas com esquizofrenia com pessoas sem essa condição por meio de experimentos variados.

2.1 Esquizofrenia e principais problemas linguísticos

A esquizofrenia é uma doença mental caracterizada pela presença de uma ou mais de uma das ocorrências a seguir: delírios, alucinações, pensamento e fala com distúrbios, comportamento motor muito desorganizado ou anormal – sintomas positivos - e sintomas negativos (DSM-5, 2013, p.87).

Os sintomas positivos refletem o excesso ou distorção do funcionamento normal, enquanto os negativos dizem respeito à ausência de características que normalmente possuem os indivíduos sem essa doença (DSM-V, 2014, p. 61). Entre os sintomas positivos destacam-se as alucinações, principalmente as verbais-auditivas (vozes); delírios, como fixação em crenças e dificuldade de se adaptar a convenções culturais; e desordem do pensamento, que se caracteriza por uma produção linguística desorganizada (KUPERBERG, 2010, p. 577).

Já os sintomas negativos mais comuns são a alogia, caracterizada pela pouca quantidade de fala ou de seu conteúdo; anedonia, que reflete a dificuldade do paciente sentir e expressar prazer, intimidade ou ter contatos sociais; embotamento afetivo; apatia e déficit de atenção (ANDREASEN, 1990, p. 616).

. Existem vários estudos que contemplam diversos problemas relativos à linguagem utilizada por pacientes com esquizofrenia. Na Fonética e Fonologia, já foi relatado, por exemplo, monotonia prosódica (COVIGTON, 2005; COMPTON, 2018); rima e aliteração na fala de pacientes (ANDREASEN, 1986); prosódia afetiva – que relata dificuldades dos pacientes em processar e expressar alguns tipos de sentimentos (LIN et al, 2018), entre outros.

Na Morfologia e Sintaxe, já foi constatado, por exemplo, simplificação sintática de pacientes (MORICE, INGRAN, 1983) e sintaxe de recepção transtornada (CONDRAY et al, 2002).

Na área da Semântica e do Léxico, já foram verificados fenômenos como a glossomania, um tipo de compulsão em utilizar palavras de um mesmo campo semântico (ANDREASEN, 1986; COVINGTON et al, 2005); bloqueio lexical (parada súbita na fala) (ANDREASEN, 1986; COVINGTON et al, 2005) e salada de palavras (COVINGTON et al, 2005).

A área da Pragmática possui quantidade bastante significativa de trabalhos (STEUBER, 2011). Entre os diversos fenômenos registrados, destacamos excesso ou pobreza de fala/conteúdo (ANDREASEN, 1986; LIDDLE et al, 2002); excesso ou pobreza de referência: excesso de referência não verbal, excesso de referência de si mesmo, excesso ou falta de informação pressuposta do interlocutor (ANDREASEN, 1986; COVINGTON et al, 2006); falta de lógica (ANDREASEN, 1986; LIDDLE et al, 2006), repetição não esperada (LIDDLE et al, 2002), perseverança em tópicos (ANDREASEN, 1986) e ecolalia (ANDREASEN, 1986).

Apesar de tantos estudos a respeito de problemas linguísticos relacionados à fala desses pacientes, foi encontrado apenas o trabalho de Cresti et al (2015) acerca de um estudo preliminar coincidente com o desta tese, isto é, sobre o padrão informacional de pessoas com esquizofrenia. Este trabalho, tal como as discussões comparativas entre pacientes e não pacientes desta tese, exigem o conhecimento de base da Teoria da Língua em Ato, a qual é discutida a seguir.

2.2 Teoria da Língua em Ato

A Teoria da Língua em Ato (CRESTI, 2000; RASO, 2012; MONEGLIA, RASO, 2014) permite a descrição linguística da estrutura informacional da fala a partir de critérios pragmáticos e prosódicos.

Trata-se de um quadro teórico no qual a unidade de referência da fala é o enunciado, o qual é definido como "a menor unidade interpretável pragmaticamente no fluxo da fala" (RASO, 2012, p. 95) e marcado prosodicamente por uma quebra percebida como terminal. No exemplo a seguir, retirado do C-ORAL-BRASIL (RASO, MELLO, 2012) é possível ouvir e perceber essa quebra terminal. É recomendável que o leitor de fato ouça os exemplos disponibilizados para acompanhar as discussões realizadas.[2]

Exemplo 1 (áudio bfammn02):

*PAU: não // tá dando a altura daquele que a Isa marcou lá / né // [3]

Em (1) há dois enunciados diferentes, marcados por barra dupla. Note-se que o primeiro é formado apenas pelo advérbio de negação "não" e é simples por ter apenas uma unidade tonal, enquanto o segundo é complexo e possui duas unidades tonais separadas por uma quebra não terminal depois de "lá". No exemplo 1, portanto, *não //* representa um enunciado, enquanto *tá dando altura daquele que a Isa marcou lá / né //* representa outro. Cada um deles realiza uma ação no mundo – seja ela qual for - denominada ato de fala. Por isso, é possível dizer que o enunciado é uma porção linguística terminada – percebida como final – e que carrega consigo algum tipo de ilocução, isto é, um ato de fala (AUSTIN, 1962).

Os enunciados são segmentados pela prosódia, a qual os divide em porções linguísticas terminadas ou não terminadas. Essa segmentação da fala é feita de modo perceptual e intuitivo por falantes de uma determinada língua. Para a Língua em Ato, uma quebra terminal indica a fronteira de uma unidade terminada, enunciado ou

---

[2]A pasta com todos os arquivos de som utilizados para exemplificação pode ser consultada. Também é possível ler o texto desses exemplos no arquivo txt disponibilizado no link abaixo ou a partir do Anexo H desta tese.
https://drive.google.com/drive/folders/10T05eZ9nqw_p1lSfobha7kJx61R5PnGW?usp=sharing

[3]Exemplo 1:
https://drive.google.com/file/d/1gfl8sN9qm4sPUGzzffEhVeim6EbtHDJS/view?usp=sharing

stanza, enquanto uma quebra não terminal indica a conclusão de suas unidades internas, isto é, de suas unidades tonais.

Essa distinção de terminalidade da quebra é registrada em transcrições de corpora orais como o C-ORAL-BRASIL (RASO, MELLO, 2012), C-ORAL-ESQ (FERRARI, ROCHA, RASO, em construção) e o C-ORAL-ROM[4] (CRESTI; MONEGLIA, 2005). Dessa forma, marca-se barra dupla "//" para uma quebra terminal – que indica a conclusão de um enunciado - e barra simples "/" – que indica o fim de uma unidade tonal – para uma quebra não terminal.

A prosódia também é responsável por marcar a função informacional da unidade tonal. Trata-se de dois níveis de análise: o segmento linguístico com parâmetros acústicos específicos, que é a unidade tonal, e sua função assumida no enunciado, que é a unidade informacional, a qual também depende de critérios posicionais – discutidos a seguir – para serem categorizadas.

A unidade informacional que veicula a ilocução é o Comentário, que é autônomo pragmaticamente, enquanto as outras unidades informacionais, que são opcionais, criam condições para que essa ilocução seja veiculada de forma eficaz. Isso significa que, para existir, um enunciado precisa, a princípio, apenas do Comentário, que é onde está localizada a ilocução. Nos exemplos 2 e 3 a seguir, é possível ouvir enunciados com apenas uma unidade tonal. Isso significa que sua porção linguística não pode ser segmentada prosodicamente em partes menores dentro do enunciado. A contraparte funcional– necessária para a determinação do padrão informacional – dessa unidade tonal única é um Comentário.

Exemplo 2 (bfammn01, 191.880 s a 193.090 s)
aí matou ele //=COM=[5]

Exemplo 3 (bfammn05, 175.620 s a 177.995 s)
eles trouxeram ela pra mim //=COM=[6]

[4] http://lablita.dit.unifi.it/coralrom/index.html.
[5]Exemplo 2 (Comentário): Disponível para escuta no link: https://drive.google.com/file/d/12LIBjT0H9VgeZmTpjBIUILU0yDz7rhZx/view?usp=sharing

[6]Exemplo 3 (Comentário): https://drive.google.com/file/d/1p1V7mMvf77ovnHamG3kz0t3i1udzDl_a/view?usp=sharing

A obrigatoriedade de haver um Comentário no enunciado completo, que é marcado pelas barras duplas, deve-se ao fato de não haver enunciado sem essa unidade informacional.

. Em conjunto com outras unidades informacionais, sistematizadas no Quadro 1, o Comentário pode formar distintos arranjos de complexidade variável – que são os padrões informacionais da fala de determinada língua.

Cada unidade tonal se associa a uma unidade informacional. A princípio, então, há um isomorfismo entre uma unidade tonal – som - e uma unidade informacional - função, exceto para unidades de Escansão, Comentários Múltiplos e Comentários Ligados (RASO, 2012, p. 113-117). Ademais, os enunciados ou *stanzas*, termo discutido na seção seguinte, se estruturam de acordo com um padrão informacional específico, isto é, de acordo com uma sequência de unidades informacionais. São essas últimas, as *stanzas*, que serão utilizadas para a comparação do padrão informacional da fala dos pacientes no capítulo V. As unidades informacionais são sistematizadas na subseção a seguir.

2.2.1 Unidades informacionais

As unidades informacionais são divididas em textuais e de auxílio dialógico. As primeiras compõem o texto do enunciado, enquanto as segundas regulam a interação entre o falante e seu interlocutor (RASO, 2012). Essas unidades são sistematizadas no Quadro 1, adaptado de Moneglia e Raso (2014, p. 490) e Cavalcanti (2020, p. 94).

No Quadro 1, é possível observar o nome da unidade informacional, sua etiqueta, que está presenta nas transcrições já etiquetadas informacionalmente; sua função e seu perfil prosódico usual, quando há. As unidades textuais, que são utilizadas nesta tese para estudar a riqueza de elaboração do padrão informacional e o quão articulada a fala dos pacientes pode ser, são: Comentário; Comentário Múltiplo; Comentário Ligado, que são unidades ilocucionárias, isto é, realizam um ato de fala; Apêndice de Comentário; Tópico, Parentético e Introdutor Locutivo. Essas unidades podem ser escandidas, conforme discutido a seguir. Já as unidades dialógicas, que medem a interatividade do enunciado e não propriamente a riqueza de sua elaboração textual, são: Fático, Incipitário, Alocutivo, Conativo, Expressivo e Conector Discursivo. Nesta tese, conforme discutido na metodologia, no capítulo III,

as unidades dialógicas são agrupadas sob um rótulo comum (AUX), pois parte da etiquetagem realizada pelo autor desta tese no C-ORAL-ESQ ainda está em processo de revisão.

Como citado na seção anterior, as unidades de Escansão (SCA), Comentários Múltiplos (CMM) e Comentários Ligados (COB) não obedecem à relação isomórfica entre unidade tonal e unidade informacional. Isso significa que não há a relação uma unidade tonal e uma unidade informacional nesses casos.

A Escansão (SCA) de uma unidade textual pode ocorrer devido a sua grande dimensão silábica, que impede o enquadramento de todo o ato locutivo em uma única unidade tonal. Isso pode ocorrer por hesitações, ênfase ou baixa diastratia, como a fala de crianças ou falantes muito jovens. O perfil prosódico dessa unidade é neutro, sem valor funcional, e o perfil da unidade informacional escandida é manifestado apenas na última unidade tonal da sequência (RASO, 2012, p. 114). No exemplo 4 a seguir, é possível ouvir um enunciado com Escansão da unidade informacional de Tópico.

Exemplo 4 (áudio bfammn06, de 287.900 s a 177.995 s):

se ele achar que tá /=SCA_r= já na hora /=TOP_r= aí eu ligo pro sior /=CMM_r= sior vai po Otaviano Neves //=CMM_r=[7]

Os Comentários Múltiplos (CMMs) ocorrem quando dois ou mais de dois Comentários estabelecem uma relação holística de interpretação para o interlocutor, separados por quebras não terminais. Assim, a realização de CMMs leva à percepção de que há uma relação de padronização retórica e prosódica entre as unidades tonais que compõem essas unidades informacionais. Dessa forma, mesmo que todos sejam ilocucionários e analisáveis autonomamente, a interpretação dos CMMs é feita em conjunto. No exemplo 5 a seguir, é possível ouvir um CMM e perceber que a ilocução é veiculada de forma holística e dependente das duas unidades tonais marcadas com CMM.

Exemplo 5 (bfammn02, 165.440 s a 167.205 s)

não /=CMM_r= nũ acredito nisso não //=CMM_r=[8]

[7] Exemplo 4 (Escansão):
https://drive.google.com/file/d/1tl1j2xQ4aw25CRxQg2RKIMIiUVLoKT0P/view?usp=sharing

[8]Exemplo 5 (Comentários Múltiplos):

Por sua vez, os Comentários Ligados (COB) são unidades informacionais ilocucionárias ligadas umas às outras por um sinal de continuidade prosódica. A presença de um COB e um COM em uma unidade terminada caracteriza essa última como uma stanza. Stanza é, portanto, uma sequência linguística terminada que, além do Comentário, também possui pelo menos um COB.

É possível observar o sinal de continuidade nas stanzas dos exemplos 6, 7 e 8 a seguir. É importante destacar que cada COB possui um sinal de continuidade e é possível perceber que a stanza ainda não chegou ao fim quando essas unidades são realizadas.

Exemplo 6[9] (bfammn06, de 372.700 s a 374.606 s):
eu também nũ tenho esse hábito /=COB= nós nũ temo //=COM=

Exemplo 7[10] (bfammn04, de 12.500 a 15.535 s):
a siora que é mãe /=COB_r= siora sabe /=COB_r= sio' pega meu filho //=COM_r=

Exemplo 8[11] (bfammn03 88.420 s a 97.522 s):
ela me fez uma compra lá /=COB_r= comprou presente pr' ocês todo /=COB_r= pos menino /=COB_r= roupa /=COB_r= e tal /=COB_r= e /=DCT_r= ea nũ me pagou /=CMM_r= também nũ cobrei /=CMM_r= e já perdoei isso há muito tempo /=CMM_r= nem converso mais não //=COM_r=

A possibilidade de haver mais de um Comentário Ligado se deve justamente a um sinal de continuidade prosódica a partir do qual podemos perceber que a stanza ainda não terminou e que se trata de uma unidade informacional ilocucionária, mas diferente do Comentário, que não possui esse sinal. Os exemplos 6, 7 e 8 ilustram

---

https://drive.google.com/file/d/1M340MmuBXg39pfH79JLe-dAtHXt9NTix/view?usp=sharing

[9]Exemplo 6 (1 COB):
https://drive.google.com/file/d/1sgELLV07_nzWMIVXaL6EDaUO3jQfhPZX/view?usp=sharing

[10]Exemplo 7 (1 COB):
https://drive.google.com/file/d/1JDhR1RnQlZt4ujZ7UYXq2kBxr-3ehm2q/view?usp=sharing

[11]Exemplo 8 (3 COB):
https://drive.google.com/file/d/15NAyGAaVv9dL3ARF6DzrVCLRN1u2FJEA/view?usp=sharing

stanzas de distintas dimensões e complexidade informacional, com 1, 2 e 5 COBs, respectivamente.

Segundo Cavalcante (2016, p. 41), stanzas são frequentes em textos nos quais a interação entre falantes é menor, de forma que existe maior elaboração semântica da dimensão textual da sequência e menor quantidade de atos de fala realizados.

As outras unidades textuais, já mencionadas nesta seção, são discutidas durante a análise da fala dos pacientes no capítulo V.

Quadro 1 – Unidades informacionais, função e perfil prosódico

Nome	Etiqueta	Função	Perfil prosódico
Unidades textuais			
Comentário	COM	Veicula a força ilocucionária do enunciado.	Núcleo prosódico com foco funcional, com características específicas da ilocução que veicula.
Comentário Ligado	COB	Sinaliza ilocução com continuidade	Sinal prosódico de continuidade.
Comentários Múltiplos	CMM	Sinalizam ilocuções convencionalizadas e distribuídas harmonicamente ao longo de pelo menos 2 CMMs	Variado.
Tópico[12]	TOP	Estabelece o âmbito de aplicação da força ilocucionária	Possui núcleo prosódico com foco funcional. Trajetória ascendente-descendente de f0 (tipo 1); trajetória ascendente de f0 (tipo 2); trajetória com valores altos de f0 no primeiro semi-núcleo, podendo ser nivelada, ascendente ou descendente no segundo semi-núcleo (tipo 3).
Apêndice de comentário / tópico	APC /	Integra textualmente a unidade da qual é apêndice	O APC tem perfil nivelado ou descendente de f0. O APT pode ter perfil semelhante ao do TOP, mas sem foco funcional.
	APT	Adiciona conteúdo textual à unidade de Tópico	Perfil prosódico conforme o tipo de Tópico.
Parentético[13]	PAR	Dá instruções sobre como deve ser interpretado o enunciado ou parte dele.	Perfil nivelado de f0, frequentemente mais baixo do que o do enunciado; velocidade de elocução mais alta que o enunciado.

[12]Tópico e Apêndice de Tópico (áudio bfammn06, de 569.320 s a 573.968 s): https://drive.google.com/file/d/1M0_qvucXTrZ9zwtvx68XpkaCFQ5silP1/view?usp=sharing

[13] Parentético (áudio bfammn01, de 264.580 s a 268.560 s): https://drive.google.com/file/d/1jdQAqNgK_mUn5gOJ8m53c7TQhZePYLBg/view?usp=sharing

Introdutor locutivo[14]	INT	Sinaliza que o que se segue tem nível hierárquico diferente daquele da enunciação (geralmente, é uma metailocução).	Perfil prosódico descendente, com f0 marcadamente mais baixo que a unidade subsequente; velocidade de elocução muito mais rápida que a do enunciado.
Unidades dialógicas			
Fático[15]	PHA	Sinaliza a abertura ou a manutenção do canal comunicativo.	Perfil sem movimento; baixa duração e intensidade; escasso conteúdo locutivo.
Incipitário[16]	INP	Sinaliza o começo do turno ou do enunciado.	Movimento ascendente-descendente e com grande variação de f0; curta duração e alta intensidade.
Alocutivo[17]	ALL	Individualiza o interlocutor. Marca coesão social.	Perfil descendente; baixa intensidade e nenhum foco.
Conativo[18]	CNT	Induz o interlocutor a cumprir ou a desistir de certa ação.	Perfil descendente; intensidade alta e curta duração.
Expressivo[19]	EXP	Fornece suporte emotivo para o ato de fala, marca coesão social.	Perfis entonacionais variados.
Conector discursivo[20]	DCT	Sinaliza continuidade de uma sequência com a anterior.	Perfil nivelado ou modulado; intensidade alta e duração longa.
Outras unidades			
Escansão	SCA	Sinaliza o conteúdo locutivo escandido de outra unidade textual, sem núcleo informacional na unidade prosódica.	Não possui perfil prosódico específico.
Tomada de Tempo[21]	TMT	Representa pausas preenchidas.	Alongamento de algum segmento.

[14] Introdutor locutivo (áudio bfammn05, de 399.180 s a 403.445 s):
https://drive.google.com/file/d/1nuUUYm56WjGp_nkSillk1CctTmQFi9BX/view?usp=sharing

[15]Fático (áudio bfammn03, de 329,740 s a 338,278 s):
https://drive.google.com/file/d/1p41Dot5L67wLFzLsDBmL5OdqkbeM9gtC/view?usp=sharing

[16]Incipitário (áudio bfammn02, de 277,680 s a 289,464 s):
https://drive.google.com/file/d/1IxIi1NBH3imO9TKKe8JcBLQ-iVz1KVhM/view?usp=sharing

[17]Alocutivo (áudio bfamm02, de 62,900 s a 65,819 s):
https://drive.google.com/file/d/1CDDLjsn4dE4r164yPZWxoYGcxGpOO5-9/view?usp=sharing

[18] Conativo (áudio bfammn03, de 284,720 s a 289,751 s):
https://drive.google.com/file/d/1zG6RVGZ88joJKTWXN5NE0KUnb8rfLEHO/view?usp=sharing

[19]Expressivo (áudio bfamdl01, de 25,640 s a 27,132 s):
https://drive.google.com/file/d/1GesuAJEhrQUj-t96FeepagA8-gvfyWJ3/view?usp=sharing

[20]Conector discursivo (áudio bfammn05, de 534,520 s a 544,059 s):
https://drive.google.com/file/d/1V3fjWQUXBKWH7oTggPmZCmXZ3tf59Dhf/view?usp=sharing

[21]Tomada de tempo (áudio bfammn01, de 13,220 s a 24,605 s):
https://drive.google.com/file/d/1wJlzOW6_T6nwD7iWFliyvDuNtJjZlbFV/view?usp=sharing

Unidade não classificada	UNC	Sinaliza uma unidade a qual não foi possível atribuir uma etiqueta	Não possui perfil prosódico específico
Unidade vazia	EMP	Utilizada quando o conteúdo informacional de uma unidade tonal não é considerado no conteúdo do enunciado. Pode ser usada em retractings e na última unidade de um enunciado interrompido.	Não possui perfil prosódico específico
i-[ETIQUETA]	i-[TAG]	Indica a suspensão momentânea de uma unidade textual por outra unidade textual	Não possui perfil prosódico específico
Unidade reportada	TAG_r	Unidade de discurso reportado	

Fonte : Adaptado de RASO, MONEGLIA, 2014 pp. 490-491; CAVALCANTE, 2020, p. 94.

Os diferentes arranjos que assumem as unidades informacionais revela o padrão informacional presente na fala. Na próxima seção, sistematizamos um estudo acerca do padrão informacional na fala de pessoas com esquizofrenia.

2.3 Teoria da Língua em Ato e fala de pessoas com esquizofrenia

A fala espontânea de indivíduos com esquizofrenia, todos italianos, já foi analisada com os pressupostos da Teoria da Língua em Ato por Cresti et al (2015). Nesse estudo, os autores utilizaram o Corpus CIPPS (DOVETTO, GEMELLI, 2012), que fornece 10 horas de interação entre quatro pacientes com esquizofrenia (A, B, C e D) e um psiquiatra. Esses indivíduos com esquizofrenia apresentavam distintos sintomas positivos e compartilhavam o sintoma de delírio, exceto A, que possuía uma espécie de pré-delírio.

Em seu estudo, de caráter preliminar, Cresti et al (2015) verificaram a relação entre prosódia e estrutura informacional, os tipos e características prosódicas das ilocuções, além da organização do padrão informacional da fala dos pacientes com esquizofrenia. Os linguistas verificaram que os indivíduos com esse distúrbio psicótico conseguem identificar enunciados por meio de quebras terminais tal como pessoas sem esquizofrenia, mas os autores notaram anomalias na fala principalmente do paciente D no que concerne à variedade de ilocuções, no padrão informacional dos enunciados e em seus perfis prosódicos.

No que tange à variação de ilocuções, os autores verificaram que o paciente D produziu 90% de asserções em seus primeiros 100 enunciados, enquanto pessoas normais geralmente produzem 50 % desse tipo de ilocução. Os linguistas também pontuam que ocasionais variações das ilocuções não atendem às demandas pragmáticas da interação e são realizadas com fenômenos típicos da fala de pessoas com esquizofrenia, tais como hiperfonia, cuja principal marca é a produção fonética com intensidade exagerada (CRESTI et al, 2015, p. 141).

Já o padrão informacional da fala de indivíduos com esquizofrenia foi menos variado do que a fala normal, e os autores relatam que nenhum Tópico foi encontrado nos primeiros 100 enunciados do paciente D, ao passo que a fala normal registra em torno de 39% de Tópicos em enunciados complexos. Essa unidade informacional, quando presente na fala do paciente D, foi encontrada apenas na tarefa de contar um conto de fadas, aparentemente, como se o paciente a tivesse decorado.

Além disso, os autores verificaram grande ocorrência de ecolalia e palilalia na fala do paciente D. A primeira são repetições de porções da fala de outra pessoa, enquanto a segunda se trata da repetição de porções da fala do próprio enunciador. Essas palilalias eram performadas com traços muito idiossincráticos, manifestados na menor variação de f0 e baixa intensidade, e resultam em um perfil prosódico que era insuficiente para veicular uma ilocução.

2.4 Comparação entre pacientes com esquizofrenia e pacientes com depressão de Steuber (2011)

Em um estudo que oferece uma visão panorâmica sobre diversos problemas linguísticos da fala de pessoas com esquizofrenia, Steuber (2011) utiliza dois corpora orais de fala espontânea, um com pacientes que sofriam de esquizofrenia e outro com pacientes que sofriam de depressão. Os corpora foram fornecidos pela empresa de prestação de serviços Verilogue[22], mas balanceados e construídos pelo autor em questão. Nesse processo, o autor selecionou 140 transcrições, bem como seus respectivos áudios, de pessoas com esquizofrenia e 50 transcrições de pacientes com depressão. O corpus de indivíduos com esquizofrenia era composto de 111.421 palavras, enquanto o de pessoas com depressão era formado por 52.347 palavras.

[22] http://www.verilogue.com/

Os participantes dos corpora tinham entre 19 a 74 anos, sendo a maioria de 35-53, dois terços de população masculina e dois terços de população caucasiana.

Já a análise dos dados consistiu em duas partes, a manual e a com o concordanciador do software Monoconc Pro 2.2[23]. A parte manual foi realizada pelo próprio linguista, que separava os fenômenos das transcrições selecionadas por presença ou frequência. Fenômenos como excesso ou pobreza de informação e salada de palavras eram analisados por presença, pois seriam de difícil contagem por *tokens*. Já fenômenos como palavras raras ou construções sintáticas eram mais fáceis de serem contados e, consequentemente, eram contabilizados por frequência. Posteriormente, Steuber enviava suas análises para cinco outros linguistas, a fim de corrigir ou corroborar sua primeira análise. Fenômenos como escolha peculiar de palavras e neologismos foram detectados pelos *types* com menor número de *tokens*.

Os resultados das comparações entre a fala de pessoas com esquizofrenia e de pessoas com depressão mostraram que a primeira é marcada por mais alterações do que a segunda. Em linhas gerais, isso converge com a literatura discutida por Steuber (2011). Com relação aos fenômenos por presença, a saber, salada de palavras e excesso ou pobreza de conteúdo ou de fala; nenhum resultado do teste qui-quadrado teve $p < 0,005$, ou seja, nenhum foi estatisticamente relevante. Concernente aos fenômenos por frequência, apenas escolha peculiar de palavras ($p=0,032$); falta de lógica ($p= 0,046$) e distração ($p= 0,020$) foram estatisticamente relevantes no teste estatístico MANOVA, fato que indica que esses fenômenos estariam mais presentes na fala de pacientes com esquizofrenia do que em pacientes com depressão.

Outro fenômeno bastante estudado na fala dos pacientes e que não é coberto pelo trabalho de Steuber são as pausas, que também são medidas e discutidas nesta tese, tanto as preenchidas quanto as silenciosas. Por isso, esses fenômenos são discutidos a seguir.

2.5 A pausa e a fala de pessoas com esquizofrenia

A pausa é uma característica comum na fala espontânea, tanto na forma de intervalos silenciosos quanto vocalizados. No intervalo silencioso, o falante não

[23] http://www.athel.com/mono.html

produz nenhuma vocalização durante algum período. Nos vocalizados, correspondentes às pausas preenchidas, o falante prolonga, por alguma razão, algum tipo de segmento geralmente sem significado lexical (ESPOSITO et al, 2006, p.542).

Apesar de claramente separadas em classes maiores, pausas silenciosas e preenchidas, não é raro que alguns autores as agrupem ao fazer generalizações a respeito da fala de pacientes com esquizofrenia. Alpert et al (1997, p.171), por exemplo, destacam que pacientes com sintomas negativos mais acentuados usualmente realizam pausas – preenchidas ou não - com maior duração, tanto em relação a pacientes com menos sintomas negativos quanto a pessoas sem esquizofrenia.

Em um antigo estudo de Feldstein e Jaffe (1963), grupos compostos por 30 pacientes com esquizofrenia e 30 sem esquizofrenia foram comparados em relação à frequência de ocorrência de pausas preenchidas. No experimento, os participantes deveriam observar quatro figuras, duas descritas como “afetivas” e duas como “não afetivas” ao longo de um minuto. Imediatamente depois, cada indivíduo deveria contar do que se lembrava da história remetida pela figura. Os resultados indicaram que pacientes com esquizofrenia realizaram mais pausas preenchidas do que o grupo controle. Para os autores, todos os indivíduos possuem uma “distribuição” de respostas comportamentais e linguísticas que são acionadas em determinados ambientes (p.778). Nos pacientes com esquizofrenia, a maior frequência de pausas preenchidas indicaria uma distribuição linguística “que é mais ampla e possui diferente probabilidades de estruturas do que a distribuição linguística de pessoas sem esquizofrenia” (p.778). Entretanto, os autores não detalham a chamada distribuição linguística e tampouco fornecem medidas em relação à duração das pausas entre os dois grupos.

Ocorre que a maior frequência de pausas preenchidas em pacientes com esquizofrenia tampouco é um consenso. Matsumoto et al (2013), por exemplo, submeteram 6 pacientes com esquizofrenia e seis pessoas sem esquizofrenia a um teste que consistia em descrever 7 placas de Rorschach vistas por meio de um espelho. Ao longo de 21 minutos, sendo 3 para cada placa, os participantes falavam o que eles viam nas placas. Foi registrado que pacientes sem esquizofrenia realizaram quantidade muito menor de pausas no experimento do que pessoas sem esquizofrenia. Segundo os autores, realizar pausas preenchidas significaria monitorar mais o discurso em relação a erros de produção e outras anomalias, e o fato de os

pacientes com esquizofrenia terem realizado menos pausas do que pessoas sem essa doença indicaria menor monitoramento discursivo desses pacientes.

2.5.1 Pausas e classes de palavras

Os estudos que comparam a distribuição gramatical das palavras diante das quais as pausas ocorrem, preenchidas ou não, são extremamente escassos. Essa escassez é grande não apenas quando se compara a frequência de ocorrência, mas também a duração. Apesar de não terem sido encontrados estudos que se debrucem sobre a possível relação de pausas e classes gramaticais de pacientes com esquizofrenia, foi desenvolvido um programa (Anexo A) a partir do qual é possível obter essas informações dos pacientes do C-ORAL-ESQ [24] e viabilizar futuros estudos que possam contemplar essa questão no caso das pausas preenchidas.

Maclay e Osgood's (1959) visavam medir, entre outras variáveis, a distribuição gramatical das pausas preenchidas e silenciosas em relação à classe gramatical das palavras diante das quais ocorriam. Para isso, os autores utilizaram uma amostra de 163 enunciados, com média 309 palavras, retirados de uma conferência universitária. Das 778 ocorrências medidas para pausas preenchidas, 418 ocorriam diante de palavras lexicais e 360 diante de palavras funcionais, isto é, as pausas preenchidas foram mais frequentes diante de palavras lexicais (p-value = 0,02 no teste do Qui-Quadrado).

Outro ponto de destaque deste estudo é que as pausas preenchidas ocorreriam mais frequentemente na fronteira de sintagmas mais extensos, enquanto pausas silenciosas na fronteira de palavras. Isso poderia indicar, argumentam, que a pausa preenchida coincidiria mais com pontos nos quais o falante parece se decidir sobre o que dizer, e que palavras de conteúdo poderiam oferecer mais dificuldade de processamento. Entretanto, os autores destacam que essa dificuldade nem sempre estaria diretamente relacionada a uma pausa preenchida, pois também foram encontrados dados nos quais o falante parece fazer escolhas linguísticas diante de pausas silenciosas.

Os achados de Maclay e Osgood's foram criticados por Cook (1971, p.36). Este último autor questionou o balanceamento dos dados utilizados pelos dois primeiros e,

[24]Naturalmente, isso também é possível com transcrições do C-ORAL-BRASIL.

além disso, chegou a resultados distintos em seu experimento. Para Cook, Maclay e Osgood's não utilizaram uma amostra suficientemente representativa e, ademais, não consideraram a frequência relativa das classes gramaticais, isto é, Maclay e Osgood's teriam assumido que todas as classes seriam igualmente frequentes em suas amostras. Após analisar a transcrição de um áudio feito durante uma entrevista com 11 falantes, Cook conclui que a probabilidade de uma pausa preenchida ocorrer antes de uma classe lexical ou funcional seria igual. O linguista também destaca que os pronomes – que ele considera uma classe lexical - tiveram frequência muito maior do que classes como substantivos e verbos antes de pausas. Além disso, o autor chama a atenção para a grande frequência de pausas preenchidas diante de conjunções, e destaca que a pausa preenchida frequentemente ocorreria no começo de uma oração ou até sua terceira palavra.

Befi-Lopes et al (2013), analisaram a duração das pausas silenciosas por classes gramaticais de 20 crianças com Distúrbio Específico da Linguagem (DEL) entre 7 e 10 anos. As autoras eliciaram os dados por meio de atividades nas quais as crianças deveriam observar quatro figuras que, juntas, poderiam formar uma história, a qual as crianças deveriam narrar. Após transcrever os dados e categorizar as palavras por classes gramaticais, foram medidas as durações de substantivos, adjetivos, verbos, conjunções, preposições e pronomes.

As linguistas perceberam que ambos os grupos realizavam pausas com menor duração diante de substantivos. Já as maiores pausas foram realizadas diante de conjunções. De acordo com as estudiosas, os substantivos são a primeira classe que as crianças aprendem e, por isso, estão bastante enraizados no léxico e são de fácil produção. Já as conjunções seriam palavras funcionais que demandariam maior complexidade linguística para elaborar a sentença, visto que essa classe gramatical, conforme argumentam, marcam relações de interdependência e dependência morfossintática que ainda são difíceis para que uma criança da idade analisada realizasse. Por isso, as pausas maiores diante de conjunções do que substantivos e outras classes gramaticais seria justificável.

Conforme será discutido na seção 3.16, a utilização do Mac-Morpho, NLTK e Pandas, viabilizou a adaptação de um etiquetador por classes de palavras treinado em textos escritos para dados do C-ORAL-BRASIL e C-ORAL-ESQ, com boa acurácia. A partir disso e de mineração de texto, já é possível encontrar pausas e suas relações com classes gramaticais com facilidade, apesar de também haver trabalho

manual envolvido para o caso da duração dessa disfluência – tanto para as preenchidas (Programa 1) quanto para as silenciosas (Programa 4).

Essas abordagens computacionais favorecem o estudo de diversos fenômenos na fala dos pacientes e podem ser úteis e auxiliares no diagnóstico da esquizofrenia em algum momento. Na seção a seguir, é discutido o trabalho de Mota et al (2017), o qual ancora sua análise na geração e análise de grafos construídos a partir de um software codificado em Java (ARNOLD et al, 2005).

2.6 Grafos e diagnóstico de esquizofrenia

Já há ferramentas computacionais para auxiliar o diagnóstico de pacientes com esquizofrenia. Em Mota et al (2017), por exemplo, os autores disponibilizam uma tecnologia que transforma frases transcritas de consultas psiquiátricas em grafos para ajudar na identificação de pessoas com essa doença. Um grafo é um tipo de representação computacional capaz de demonstrar conexões de possíveis sequências, neste caso, linguísticas.

Nos grafos de Mota et al, as palavras são representadas por nós ligados por arestas. Para esses autores, quanto menos conectados são esses nós, maior a quantidade de sintomas negativos que podem apresentar esses pacientes, isto é, há uma correlação negativa entre conexão de nós e sintomas negativos, isto é, quanto menos conexão, mais sintomas negativos.

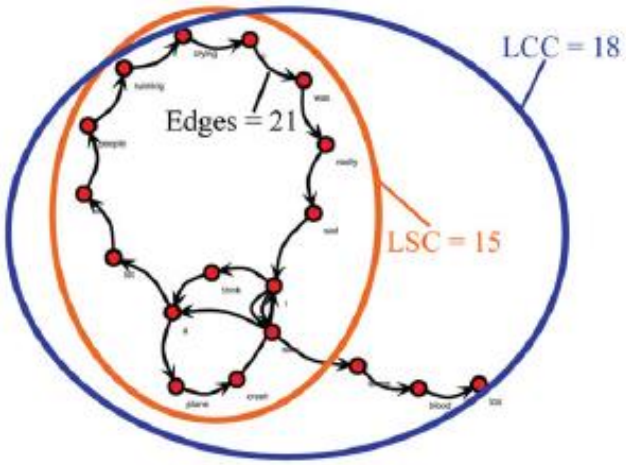
Para investigar como a fala desses pacientes poderia ajudar no diagnóstico, os autores selecionaram 21 pacientes, sendo 11 com esquizofrenia e 10 com distúrbio bipolar, além de 21 indivíduos para grupo controle. Neste artigo, os autores estavam particularmente interessados em descobrir como um discurso randômico, discutido a seguir, poderia ajudar na classificação de sintomas negativos.

Para fazer essa comparação, os autores selecionaram excertos de 30 segundos de cada um dos pacientes, os quais deveriam fazer I) o relato de um sonho novo ou antigo II) o relato de algo que aconteceu de um dia anterior ao sonho III) um relato da última coisa de que se lembravam IV) um relato baseado – ou uma pequena história – a partir de uma imagem de conteúdo afetivo forte, que poderia ser positiva,

neutra ou negativa. Depois, os áudios eram transcritos e transformados em grafos por um software[25].

A partir de cada grafo, foi possível calcular a quantidade de nós – que representam palavras – conectados por algum caminho (LCC), a quantidade de arestas (E) que representam a conexão em si entre um nó e outro, e a quantidade de nós conectados direta ou indiretamente por caminhos recíprocos, isto é, o nó A chega ao nó B e B chega ao nó A. Isso pode ser visualizado na Figura 1, extraída de Mota et al (2017, p.127), que representa o trecho *I saw a plane crash, I think, a lot of people running, crying. It was really sad. I saw some blood too.*

Figura 1 – Exemplo de grafo gerado a partir do trecho transcrito



Fonte: Mota et al (2017, p.117)

Na Figura 1 há um total de 18 nós (LCC=18), 21 arestas (E = 21) e 18 nós ligados por um caminho recíproco (LSC = 18).

Depois, foram gerados 1000 grafos aleatórios a partir de cada grafo inicial, de modo a tentar simular matematicamente uma possível desorganização da fala desses pacientes ao comparar com os grafos originais. Esses grafos aleatórios aparentam funcionar como um modelo n-gram, discutido na seção 3.16, que separa combinações aleatórias de palavras de algum texto – em sequências n (n= 1, n= 2, n =3, etc) e transforma essas palavras em vetores, que são representações matemáticas de uma estrutura de dados dispostas em matrizes com o intuito de representar uma informação que,neste caso, são os trechos transcritos dos pacientes.

25

A partir disso, foi possível calcular a diferença entre a quantidade de nós totais aleatórios, com conexões recíprocas (LCSz) ou não (LCCz), dos originais. Se essa diferença, marcada em escala *z-score*, fosse até 2 pontos de desvio padrão, a fala era considerada mais desorganizada, pois se assemelharia mais aos grafos randômicos, ao passo que uma diferença maior implicaria uma fala mais organizada, pois mostraria justamente que o trecho em questão se distanciava dos grafos aleatórios.

Os resultados mostraram que 64% dos pacientes com esquizofrenia apresentaram essa desorganização, enquanto isso ocorria com 5% do grupo controle. Em suma, os pacientes demonstraram menos nós totais (LCC) e recíprocos (LSC) do que pessoas com distúrbio bipolar e o grupo controle, além de menor diferença entre os grafos aleatórios, isto é, apresentaram menor índice LCCz, que mede a diferença de nós totais para os aleatórios, e menor LSCz, que mede a diferença entre os nós reciprocamente conectados originais dos aleatórios.

2.7 Prosódia emocional e esquizofrenia

Outra importante contribuição que pode auxiliar no diagnóstico da esquizofrenia é o trabalho com a chamada prosódia emocional ou afetiva. Segundo Ferreira-Neto (2006, p. 26), a prosódia emocional de uma frase se caracteriza “pela variação de tons abaixo de 1KHz em pontos da frase, de maneira a estabelecer uma curva melódica bem definida que os ouvintes associam a diferentes estados emocionais”. Trata-se de uma forma de comparar parâmetros acústicos variados, tais como frequência fundamental, intensidade e duração, na ocasião de manifestação de distintas emoções, como raiva, medo, tristeza e felicidade.

A partir de experimentos e softwares variados, como o ExProsodia (FERREIRA-NETO, 2010), é possível transformar esses dados gravados e analisá-los estatisticamente para aferição de medidas de tendência central e dispersão, de modo a obter padrões recorrentes na fala que podem ser utilizados para fazer comparações entre grupos diferentes de participantes.

Nesse sentido, o trabalho de Jorge (2018, 2020) obteve conquistas importantes. Jorge comparou a fala de 32 sujeitos com esquizofrenia com 16 controles em quatro atividades distintas: anamnese, relato de momentos felizes e tristes, descrição de obras visuais e leitura.

Na anamnese, que consistia em um relato do paciente sobre sua rotina diária e cuidados médicos, pacientes com esquizofrenia registraram menor assimetria da média de frequência fundamental (p = 0,017) e tom médio (média da frequência fundamental) mais baixo e com menor dispersão (p =0,013). No relato de algum momento feliz ou triste, os pacientes manifestaram menor variação da curva da frequência fundamental, a qual acompanhou, de forma geral, o tom médio da frequência. Os sujeitos controle, por outro lado, apresentaram diversos momentos com picos de frequência, os quais indicariam ocasionais momentos de foco e ênfase, segundo Jorge. Na atividade que consistia na descrição de alguma obra visual, os pacientes também manifestaram menor variação da frequência e poucos mementos de foco e ênfase. Por fim, os pacientes também apresentaram grande achatamento da curva de frequência na leitura de um trecho quando comparados com pessoas sem esquizofrenia (p = 0,002).

3 METODOLOGIA

Este capítulo contempla os procedimentos requisitados para a comparação entre pacientes do C-ORAL-ESQ e de pessoas do C-ORAL-BRASIL, além da metodologia empregada para descrever uma amostra representativa de pacientes com esquizofrenia do primeiro corpus. Esquematicamente, a comparação entre os dois copora é discutida entre as seções 3.1 a 3.11, enquanto a parte descritiva nas seções subsequentes.

As considerações iniciais são feitas na seção a seguir. Na seção 3.2, são apresentadas as perguntas de pesquisa deste trabalho, seguidas pelas hipóteses na seção 3.3. Um breve panorama da linguagem Python, que sustenta toda a metodologia e tecnologia produzida para este trabalho, é apresentado na seção 3.4 e, ocasionalmente, ao longo do texto quando necessário. Os corpora utilizados na comparação são detalhados na seção 3.5, assim como os procedimentos para montagem do corpus de pesquisa na seção 3.6– uma vez que o C-ORAL-ESQ ainda está em fase de construção e revisão. Dados sociolinguísticos e clínicos dos participantes da comparação são apresentados nas seções 3.7 e 3.8 e a prescrição de medicamentos dos pacientes na seção 3.9. A montagem do subcorpus para comparação de padrões informacionais e os procedimentos e parâmetros para comparação são apresentados nas seções 3.10 e 11, respectivamente.

Na seção 3.12, são apresentados os procedimentos para descrição lexical e de medidas da fala do C-ORAL-ESQ, enquanto os dados sociolinguísticos e clínicos desses pacientes – que também detalham suas prescrições de medicamentos - são explicitados nas seções 3.13. Na seção 3.14, são detalhados os procedimentos para descrição de medidas da fala e disfluências dessa etapa – muitos deles utilizados também na comparação. Posteriormente, são discutidos os procedimentos de normalização ortográfica, que é base para descrição lexical, na seção 3.15. A seguir, na seção 3.16, são explicitados detalhes do treinamento de um etiquetador de classes de palavras criado e adaptado para o C-ORAL-ESQ e C-ORAL-BRASIL por meio da NLTK, com treinamento no Mac-Morpho. Essa etiquetagem viabiliza o detalhamento do léxico, quer pelo DELAF, com informações morfológicas das classes e lematização, quer pelo web scraping na VerboWeb, que enriquece a descrição da fala dos pacientes do ponto de vista semântico e sintático, procedimentos esses descritos na seção 3.17.

3.1 Considerações iniciais

Conforme já discutido no capítulo de pressupostos teóricos, já foram registradas diferenças linguísticas estatisticamente significativas entre pacientes com esquizofrenia e sem esquizofrenia.

No que concerne ao padrão informacional, com exceção do estudo preliminar de Cresti et al (2015, cf. seção 2.3), não há outro estudo quantitativo, baseado em dados de fala espontânea, que compare a distribuição das unidades informacionais que compõem a fala entre pacientes com esquizofrenia e sem essa condição.

O presente trabalho não poderia preencher completamente essa lacuna. Apesar de viés quantitativo-estatístico e baseado em corpora de fala espontânea representativos de suas populações, C-ORAL-BRASIL e C-ORAL-ESQ possuem contextos de interação diferentes. O C-ORAL-BRASIL não é um corpus controle do C-ORAL-ESQ e o gênero consulta psiquiátrica deste segundo não consta na variada arquitetura do primeiro.

Além disso, é verificado que alguns áudios utilizados do C-ORAL-BRASIL possuem predomínio de monólogo muito superior ao do C-ORAL-ESQ, pois existem menos intervenções no turno monológico de quem conta uma história em alguns áudios do C-ORAL-BRASIL do que na fala de pacientes do C-ORAL-ESQ em trechos mais monológicos.

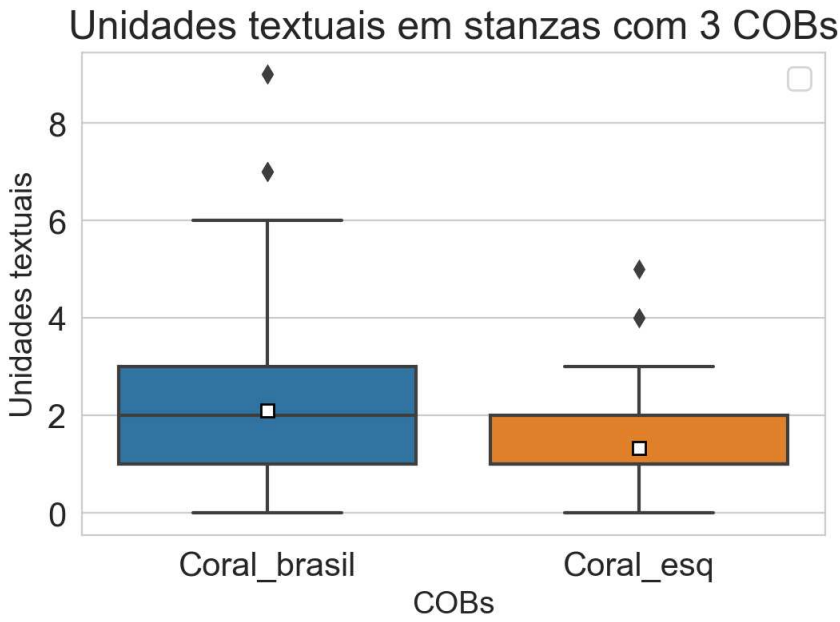
Se são corpora com contextos de interação diferentes, como localizar o que seria comum e, a partir daí, encontrar o que seria comparável?

Este trabalho propõe que a stanza e a estrutura informacional que a compõe, principalmente a distribuição de suas unidades textuais, seja esse ponto de partida. Já foi verificado que a stanza é característica principalmente de textos monológicos, nos quais pode adquirir complexidade notável (CAVALCANTI, 2020, p. 58). Por isso, utilizar a stanza como base para a comparação entre padrões informacionais poderia ser uma forma de compará-los.

Considere-se o exemplo abaixo, baseado em 92 stanzas com 3 COBs, sendo 46 do C-ORAL-ESQ e 46 do C-ORAL-BRASIL. No eixo das ordenadas está plotada a quantidade de unidades textuais por stanza. No eixo das abscissas, estão representadas as amostras dos dois corpora. É possível observar que o C-ORAL-BRASIL possui mais unidades textuais em stanzas dessa magnitude do que o C-

ORAL-ESQ, conforme é possível verificar tanto pela posição mais elevada da média e da mediana, representadas pelo quadradinho branco e pela linha transversal que corta as caixinhas, respectivamente.

Figura 2 – Unidades textuais em stanzas com 3 COBs



Fonte: Elaboração do autor.

Dessa forma, é possível sugerir que stanzas com a mesma quantidade de Comentários Ligados nos dois corpora poderiam apresentar padrões informacionais diferentes.

3.2 Perguntas de pesquisa

As perguntas de pesquisa deste trabalho são: qual é a distribuição das unidades textuais nos dois grupos em enunciados que apresentam o mesmo número de COBs? A frequência de ocorrência de cada unidade textual nessas stanzas tem diferença estatística relevante? Quantos padrões únicos possuem essas stanzas? Quantas vezes ocorrem?

Com relação às disfluências da fala a serem analisadas neste trabalho (cf. seção 3.11.6), as perguntas de pesquisa são: pacientes com esquizofrenia realizam pausas preenchidas e silenciosas com duração maior do que os indivíduos sem

esquizofrenia? Há mais retractings, palavras retratadas, escansões e enunciados interrompidos na fala desses pacientes ou de pessoas sem esquizofrenia? Quais as características fonológicas desses retractings, tais como quantidade sílabas e classificação do segmento inicial?

3.3 Hipóteses nulas e hipóteses alternativas

Hipótese nula 1: Não existe diferença estatística relevante entre a quantidade de unidades informacionais textuais entre os dois corpora.
Hipótese alternativa 1: Existe diferença estatística relevante entre a quantidade de unidades informacionais textuais entre os dois corpora. Se isso ocorre, então $p < 0,05$[26] nos testes U de Mann Whitney ou T de Student, de acordo com a distribuição da amostra.
Hipótese nula 2: Não existe diferença estatística relevante entre as disfluências encontradas nos dois grupos.
Hipótese alternativa 2: Existe diferença estatística relevante entre as disfluências discutidas. Dessa forma, $p < 0,05$.

3.4 Linguagem Python

Esta metodologia utiliza amplamente a linguagem Python, versão 3.8.5, e várias bibliotecas externas e integradas (chamadas *built-in)* a essa linguagem. As bibliotecas mais comuns foram Pandas e Numpy (MCKINNEY, 2010), para análise estatística, contagens diversas e operações em dataframes; Seaborn (WASKOM, M. et al, 2017) Matplotlib (HUNTER, 2007) e Plotly (INC, 2015) para visualizações; Natural Language Toolkit – NLTK (BIRD et al, 2009) e Spacy (HONNIBAL; MONTANI, 2017) para processamento de linguagem natural, etiquetagem por classe de palavras e outros; *re* de expressões regulares e mineração de texto, entre outras.

Todos os procedimentos e códigos descritos aqui foram realizados em um computador do tipo notebook da marca Dell, processador intel Core i5 com 8 gigas de

[26]Quando estatisticamente relevante e de valor muito baixo, como 2,66E+08, isso será representado apenas como $p < 0,01$ por uma questão didática.

memória Ram. Os códigos foram feitos principalmente no Spyder, outro ambiente Python, por possuir acesso visual a variáveis e ser gratuito. Ocasionais visualizações e tabelas são disponibilizados por link em notebooks do Google Colab – onde o leitor também pode executar a maioria dos programas disponibilizados no Anexo sem necessidade de instalação.

3.5 Corpora utilizados

Os corpora utilizados neste trabalho, como já mencionado, foram o C-ORAL-ESQ, representativo da fala de pacientes com esquizofrenia, e C-ORAL-BRASIL, representativo do português brasileiro falado. Por representativo entende-se um corpus que contemple a distribuição estratificada de sua população e de fenômenos linguísticos para os quais esse corpus visa a representar. Isso significa que as amostras presentes nele presentes devem contemplar a variedade linguística e social de sua população (BIBER, 1993, 242-246). No caso dos corpora supracitados, para além de quantidade de palavras ou de sujeitos participantes, é necessário considerar, por exemplo, questões sociolinguísticas como nível educacional; idade e sexo; além de questões clínicas no caso do CORAL-ESQ, como pontuação em escalas diversas de sintomas positivos e negativos.

3.5.1 CORAL-ESQ

O corpus C-ORAL-ESQ, em fase de compilação, se insere no projeto “A compilação do corpus C-ORAL-ESQ e o estudo da fala de pacientes com esquizofrenia”, coordenado pelo professor Tommaso Raso e subcoordenado pelo professor João Vinícius Salgado, da Universidade Federal de Minas Gerais. Trata-se de um corpus que visa ser representativo da fala espontânea de pacientes com esquizofrenia, os quais são gravados durante atendimento médico psiquiátrico em um hospital psiquiátrico em Belo Horizonte.

Arquitetonicamente, o C-ORAL-ESQ conterá em torno de 40 gravações com média de 1.500 palavras cada, e um total de cerca de 30.000 palavras. Além de médico e paciente, possíveis acompanhantes podem estar presentes nas consultas. A participação nas consultas é de caráter voluntário e todos os participantes da

consulta assinam um Termo de Consentimento Livre e Esclarecido, bem como fornecem dados sociolinguísticos como idade, sexo, ocupação, escolaridade e origem. Os dados sociolinguísticos e algumas informações clínicas desses participantes são apresentados na seção 3.8.

O minicorpus do C-ORAL-ESQ, ainda em compilação, consistirá de 20 textos, com o total aproximado de 15 mil palavras, etiquetado informacionalmente segundo os critérios da Teoria da Língua em Ato (cf. seção 2.3).

3.5.2 C-ORAL-BRASIL

O C-ORAL-BRASIL I (RASO; MELLO, 2012) é um corpus oral de fala espontânea do português brasileiro, com foco na diatopia do português falado em Belo Horizonte, capital de Minas Gerais. Esse corpus possui grande variação diafásica, pois são muitos contextos nos quais diálogos, monólogos e conversas informais entre três ou mais participantes foram gravadas. O C-ORAL-BRASIL I possui 208.130 palavras divididas em 139 interações com cerca de 1500 palavras cada uma. Tal como o C-ORAL-ESQ, o C-ORAL- BRASIL fornece dados sociolinguísticos dos participantes. Da mesma forma, o C-ORAL-BRASIL compartilha os pressupostos de transcrição e segmentação, conforme já discutido nesta tese.

O minicorpus do C-ORAL-BRASIL possui 6 conversas com mais de dois participantes, 7 diálogos e 7 monólogos, em um total de 20 textos com 31.318 palavras (RASO, MELLO, 2012, p. 220-221). Convém destacar que esse minicorpus, tal como o C-ORAL-BRASIL I, também apresenta grande variação de diafásica. Essa variedade diafásica não é registrada no C-ORAL-ESQ, pois este corpus possui um contexto de enunciação muito circunscrito à interação médico-paciente e ocasionais acompanhantes presentes nas consultas psiquiátricas.

3.6 Procedimentos para montagem do corpus de pesquisa

A montagem do corpus de pesquisa para esta tese consistiu em gravação e transcrição, alinhamento entre texto / som, etiquetagem informacional e codificação para extração e tratamento dos dados. Os dois primeiros processos contaram também com a coloboração da equipe do LEEL– Laboratório de Estudos Empíricos da Linguagem. A etiquetagem dos seis áudios utilizados foi primeiramente feita pelo autor

desta tese e está em processo de revisão pela equipe do LEEL. Para a comparação realizada neste trabalho, são utilizados 3 áudios já revisados e 3 áudios com etiquetagem informacional preliminar.

3.6.1 Gravação

As gravações para este trabalho, também presentes no C-ORAL-ESQ, foram realizadas em um hospital psiquiátrico de Belo Horizonte. Havia o consentimento prévio dos pacientes e ocasionais acompanhantes, além da equipe médica envolvida. Os equipamentos utilizados nas gravações são dois microfones de lapela omnidirecionais Sennheiser EK100/SK100 conectados a um gravador Taskam DR-100 ou Marantz PMD-660. As gravações são feitas em formato estéreo, salvas em formato *wav* e possuem taxa de amostragem de 44.100 Hz. É importante destacar que as gravações do C-ORAL-ESQ foram interrompidas desde o início da pandemia de coronavírus e isso causou diversos empecilhos para o andamento do projeto, tais como atraso na disponibilização de dados, alinhamento e etiquetagem.

A partir desses áudios é realizada a transcrição, conforme os critérios da Teoria da Língua em Ato (cf. MELLO, RASO et al, 2012, pp.125-145). Depois, é feito um alinhamento entre texto e som. Nesta tese, foram utilizados arquivos alinhados e gerados pelo WinPitch (MARTIN, 2004).

Posteriormente, o autor deste trabalho realizou a etiquetagem informacional preliminar dos áudios dos pacientes. Neste último processo, cada unidade tonal da transrição recebe uma etiqueta que marca sua função informacional (cf. Quadro 1, na seção 2.2.1). Trata-se de um processo que requisita a análise visual e sonora de parâmetros prosódicos das unidades tonais e que é inteiramente manual (cf. CAVALCANTE, 2020, pp. 92-93).

3.6.2 Extração dos dados

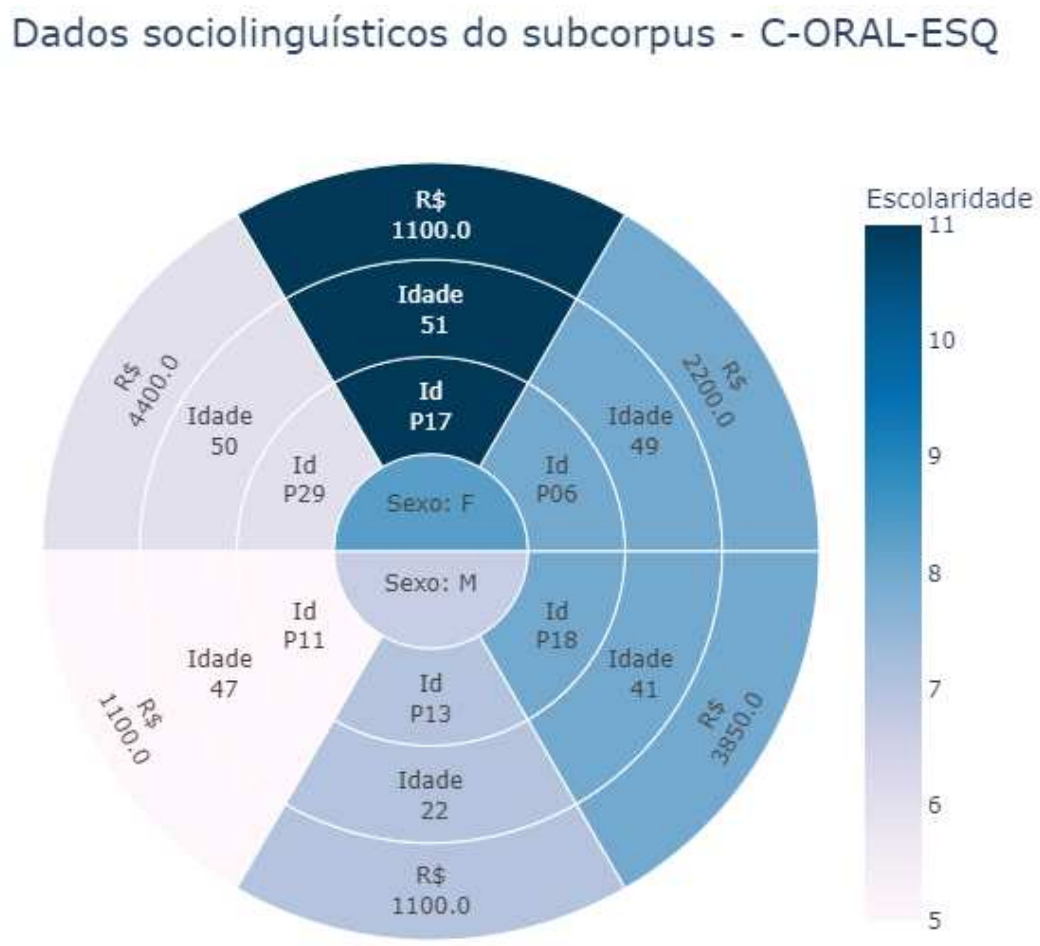
Os dados foram extraídos e tratados com linguagem Python por meio dos programas disponibilizados no Anexo desta tese, assim como outras codificações posteriores. Para gerar um subcorpus já balanceado de pesquisa, assim como todos

os resultados discutidos nos capítulos IV e V, é necessário utilizar o Programa 1[27] (cf. Anexo A). Os dados sociolinguísticos e clínicos dos pacientes e participantes desse subcorpus são discutidos na seção a seguir.

3.7 Dados sociolinguísticos das transcrições utilizadas para comparação

No conjunto de informantes do C-ORAL-ESQ, há 3 homens e 3 mulheres, com idade de 22 a 51 anos. Essas pessoas declararam renda de 1100 a 3850 reais e a escolaridade predominante é de 7 a 9 anos, conforme é possível visualizar pelos tons de azul do termômetro de cor na Figura 3.

Figura 3 – Dados sociolinguísticos do subcorpus do C-ORAL-ESQ



Fonte: Elaboração do autor.

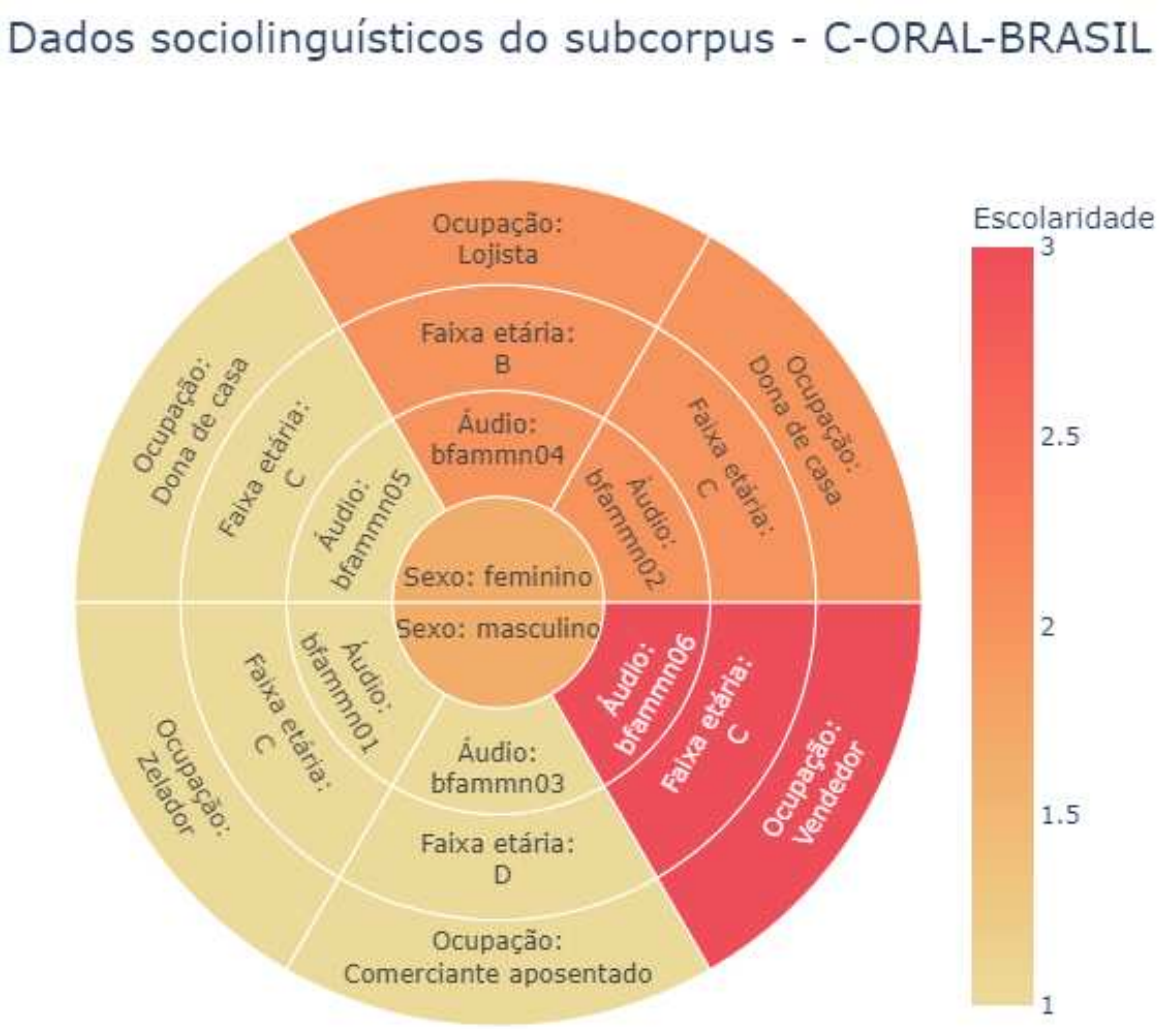
[27]Vídeo para ilustrar como é feita a extração: https://www.loom.com/share/15a2516728224fcc968cbeebc6093531

Já a Figura 4 apresenta os dados sociolinguísticos do subcorpus feito a partir do minicorpus do C-ORAL-BRASIL. É possível observer que há três participantes do sexo feminino e três do masculino. As faixas etárias descritas a seguir seguem o padrão estabelecido em Raso e Mello (2012, p. 67) para os dados sociolinguísticos desse corpus.

Dois dos três homens possuem entre 41 e 60 anos (faixa etária C), e outro mais de 60 (faixa etária D). Já a faixa etária das mulheres vai de 26 a 60 anos, sendo uma participante na faixa etária B, de 26 a 40 anos, outra na C e a última na D, acima de 60 anos

Pelo termômetro de cor, observa-se que a maioria dos participantes têm o primário incompleto (1) e apenas 1 possui curso superior e trabalha na área (3).

Figura 4 – Dados sociolinguísticos do subcorpus do C-ORAL-BRASIL

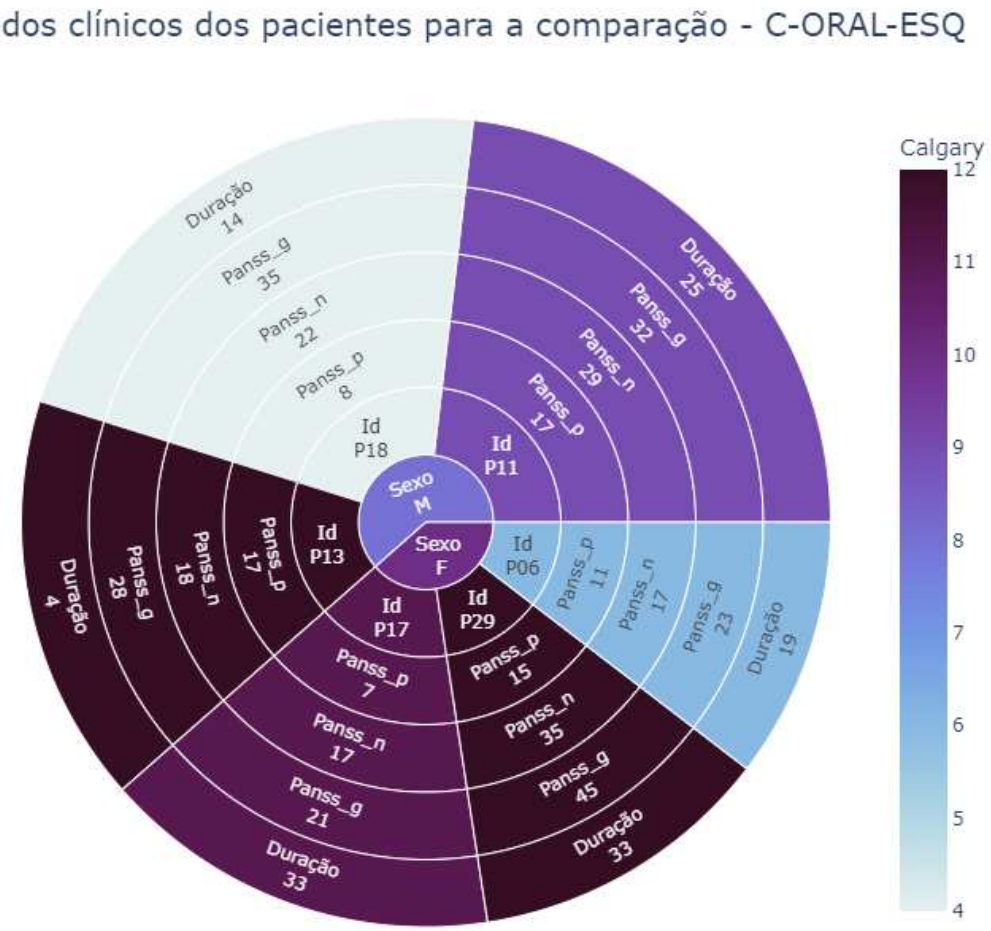


Fonte: Elaboração do autor.

3.8 Dados clínicos de pacientes do C-ORAL-ESQ para comparação

Alguns dados clínicos dos pacientes estão plotados na Figura 5. Além de sexo e identificação (id), é possível visualizar a pontuação desses indivíduos nas Escalas de Sintomas Positivos (Panss_p), Negativos (Panss_n) e Gerais (Panss_g), assim como o tempo que esses participantes do C-ORAL-ESQ convivem com a esquizofrenia (Duração). No termômetro de cor, é possível visualizar a pontuação de cada um desses pacientes na Escala Calgary, que mede o nível de depressão desses indivíduos, tal como discutido a seguir.

Figura 5 – Pontuação em escalas de sintomas e duração da esquizofrenia dos pacientes



Fonte: Elaboração do autor.

A PANSS consiste na avaliação de três escalas que distinguem sintomas positivos dos negativos e uma terceira em psicopatologias (cf. KAY et al, 1989). Cada sintoma é avaliado em uma escala de 1 a 7, sendo 1 considerado ausente e 7 considerado extremo. Posteriormente, é feita uma soma com a pontuação correspondente a cada escala.

A PANSS de sintomas positivos avalia delírios; desorganização conceitual; compulsão alucinatória; excitação; grandeza; desconfiança e hostilidade. Na Figura 5, os pacientes têm de 8 a 17 pontos nessa escala.

Já a PANSS de sintomas negativos mede o embotamento afetivo; retraimento emocional; contato pobre; retraimento social; dificuldade em pensamento abstrato; falta de espontaneidade e fluxo da fala e pensamento estereotipado. Na Figura 5, os participantes do C-ORAL-ESQ têm de 17 a 35 pontos nessa escala.

Por sua vez, a escala PANSS de psicopatologias afere preocupação somática; ansiedade; sentimentos de culpa; tensão; maneirismos e postura; depressão; retardo motor; falta de cooperação; pensamento incomum; desorientação; déficit de atenção; falta de julgamento e crítica; distúrbios de volição; falta de controle de impulsos; preocupação e esquiva social. Na Figura 5, os indivíduos com esquizofrenia possuem de 21 a 45 pontos nessa escala.

A Escala Calgary de Depressão para Esquizofrenia (ADDINGTON et al, 1993) mede o nível de depressão dos pacientes com esquizofrenia por meio de 9 perguntas, as quais são feitas e avaliadas por um psiquiatra de acordo com as respostas e observações durante a entrevista. Essas perguntas avaliam, respectivamente: depressão do humor; desesperança; autodepreciação; referências de acusação de culpa; culpa patológica; depressão matutina; despertar precoce; suicídio e depressão observada. O paciente possui 4 alternativas para resposta, a saber, 0 (ausente), 1 (leve), 2 (moderado), 3 (severo). Posteriormente, é feito um escore geral com o somatório das respostas. Dessa forma, é possível destacar que, quanto mais pontos o paciente apresenta nesta escala, mais deprimido ele é. Na Figura 5, os pacientes representados com as cores mais escuras são os mais deprimidos.

3.9 Prescrição dos pacientes

A prescrição de medicamentos utilizados pelos pacientes pode ser visualizada na Tabela 1 a seguir.

Tabela 1 - Prescrição de medicamentos dos pacientes do subcorpus de comparação

Id	Áudio	Prescrição
P06	MED_007	Risperidona 3mg/d + Clorpromazina 25mg/d + Fluoxetina 20mg/d + Clonazepam 2mg/d
P29	MED_008	Clozapina 500mg/dia + Haloperidol 10mg/dia
P11	MED_013	Clozapina 600mg/dia + Haloperidol 5mg/dia + Prometazina 100mg/dia + Clorpromazina75mg/dia + AVP 750mg/dia + Biperideno 2mg/dia
P13	MED_015	Olanzapina 10mg/dia
P17	MED_019	Olanzapina 25mg/dia + AVP 500mg/d + Fluoxetina 20mg/d + Clorpromazina 100mg/d
P18	MED_020	Quetiapina 200mg/dia

Fonte: Elaboração do autor

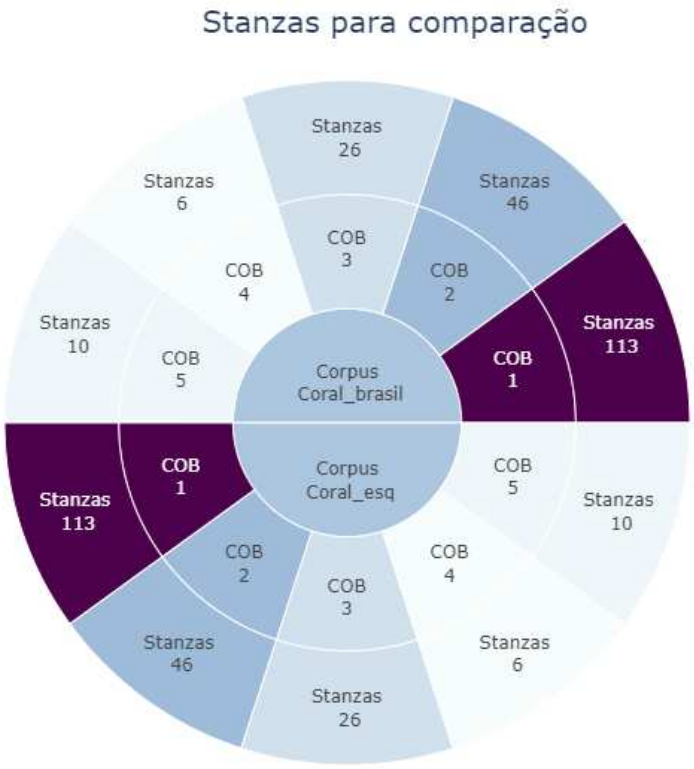
3. 10 Montagem do subcorpus para comparação de padrões informacionais

O subcorpus para comparação de padrões informacionais foi montado com 6 transcrições do C-ORAL-ESQ e 6 transcrições de monólogos do C-ORAL-BRASIL (Programa 1, Anexo A). O programa seleciona as stanzas e as agrupa de acordo com seu número de COBs. Assim, stanzas com 1 COB do C-ORAL-ESQ ficam juntas com as de 1 COB do C-ORAL-BRASIL e assim por diante.

Neste caso, foram feitas amostras aleatórias. Dessa forma, stanzas de 1 COB são comparadas com stanzas de 1 COB; as de 2 COB com as de 2 COB e assim sucessivamente. O resultado final é um subcorpus balanceado por stanzas que possuem igual número de COBs, tal como pode ser visualizado a seguir.

Na Figura 6, as amostras mais representativas estão em cores mais escuras. É possível observar que as mais representativas e que devem ser levadas em conta nas discussões aqui empreendidas são certamente as de 1 a 3 COBs devido à quantidade de stanzas disponíveis para comparação. No processo de balanceamento, foram descartadas amostras com menos de 5 COBs por corpus devido a sua pouca relevância estatística.

Figura 6 - Quantidade de stanzas disponíveis para comparação



Fonte: Elaborado pelo autor.

3.11 Procedimentos para comparação entre os dois corpora

Esta seção detalha os procedimentos empregados para realizar diversas comparações relativas aos padrões informacionais encontrados nos dois corpora.

3.11.1 Padrões informacionais

A contagem dos padrões informacionais é feita em duas partes. Na primeira é feita uma contagem de quantas vezes ocorre determinado padrão, operação cujo objetivo é dar uma visão geral da frequência desses padrões nos dois corpora. A Tabela 2 a seguir ilustra esse procedimento com os 5 padrões mais frequentes no subcorpus de comparação.

Posteriormente, é realizada outra contagem, dessa vez dada pela frequência de cada unidade informacional textual dos padrões contabilizados anteriormente.

Dessa forma, é possível ter uma visão geral da ocorrência dos padrões informacionais e também um estudo mais detalhado da ocorrência de cada unidade que os formam

Tabela 2 - Exemplo de padrões informacionais mais frequentes nos corpora

Corpus	Padrões	Frequência
Coral_esq	=COB= =COM=	68
Coral_brasil	=COB= =COM=	25
Coral_esq	=COB= =COB= =COM=	18
Coral_esq	=AUX= =COB= =COM=	11
Coral_brasil	=COB= =COB= =COM=	9

Fonte: Elaboração do autor.

3.11.2 Como é feita a comparação

Conforme já ponderado, a comparação nesse subcorpus tem por base a quantidade de COBs por stanzas. A partir disso, é possível contabilizar a quantidade de etiquetas que circundam o Comentário Ligado, ora enriquendo a stanza textualmente por uma unidade textual, ora referindo-se à própria interação por meio de uma unidade dialógica.

No caso dos padrões informacionais deste trabalho são de especial interesse as etiquetas de unidades textuais, pois a soma de cada uma delas revela o quão elaborado pode ser esse padrão informacional, de forma que é possível comparar os dois grupos desta tese. Além das unidades textuais, também são comparadas medidas da fala e disfluências. As variáveis a serem comparadas entre os dois grupos e discutidas a seguir são:

i. Unidades textuais totais;
ii. Unidades textuais individuais;
iii. Unidades dialógicas;
iv. Disfluências: quantidade de retractings, classificação fonética e fonológica do segmento do segmento inicial do retracting; palavras retratadas; quantidade de enunciados interrompidos; duração de pausas preenchidas em Tomadas de Tempo; duração de pausas silenciosas.

3.11.3 Unidades textuais totais

A partir do subcorpus já balanceado (cf. seção 3.10), é possível realizar as contagens. No caso das unidades textuais totais, são somadas todas as unidades textuais, exceto Comentário Ligado, que é o parâmetro para comparação, e Escansão, a qual representa a própria unidade textual escandida em mais de uma unidade tonal, isto é, trata-se de uma extensão da própria unidade textual em questão, à esquerda (cf. seção 2.2.1).

As unidades textuais somadas são: Comentário, Apêndice de Comentário, Comentários Múltiplos, Tópico, Apêndice de Tópico, Parentético, Introdutor Locutivo. Nessas contagens, é possível ver a soma, média, mediana, desvio padrão, valor mínimo e valor máximo da distribuição. Todas essas medidas nem sempre são necessárias nas discussões propostas, de modo que ora são utilizadas no corpo do texto, ora podem ser visualizadas em um notebook do Google Colab.

Tabela 3 – Exemplo de como é feita contagem de unidades textuais totais

COB	**corpus**	**sum**	**mean**	**std**	**median**	**min**	**max**
1	Coral_brasil	215	1.84	1.19	2.0	0	6
1	Coral_esq	143	1.22	0.63	1.0	0	3
2	Coral_brasil	117	2.54	1.82	2.0	0	7
2	Coral_esq	62	1.35	0.82	1.0	0	3

Fonte: Elaboração do autor.

Essas operações são utilizadas para análise de todas as unidades informacionais, de acordo com a quantidade de COBs do enunciado em questão. A seguir, é apresentado um exemplo do resultado dessas operações com 1 e 2 COB na coluna que soma todas as unidades textuais.

Quanto maior a frequência do conjunto dessas unidades textuais em uma mesma stanza, maior a complexidade desses padrões. No entanto, essa comparação revela apenas a quantidade do conjunto das unidades textuais. Para analisar a complexidade na elaboração desses padrões, é necessário verificar a quantidade de ocorrência de cada uma delas. Os procedimentos para isso são expostos na seção a seguir.

3.11.4 Unidades textuais individuais

Após analisadas em conjunto, é necessário contabilizar a ocorrência de cada uma das unidades textuais separadamente, de acordo com os mesmos critérios estabelecidos na subseção 3.11.3. As unidades textuais analisadas separadamente são: Comentários, Apêndice de Comentário, Comentários Múltiplos, Tópico, Apêndice de Tópico, Parentético e Introdutor Locutivo.

3.11.5 Unidades dialógicas

Neste trabalho, as unidades dialógicas são contabilizadas em uma coluna única para o C-ORAL-ESQ. Isso se deve ao fato de que a etiquetagem dessas unidades foi marcada com um rótulo único de =AUX= em alguns dos áudios etiquetados para esta tese. Após futuras revisões, é possível que esse rótulo seja mudado para o símbolo correspondente de cada unidade, com suas devidas contagens de frequência. Isso não altera o funcionamento do Programa 1 (Anexo A), que também conta as unidades dialógicas separadamente quando elas existem.

Como a etiquetagem do C-ORAL-BRASIL já possuía os rótulos específicos para unidades dialógicas, o código também prevê que cada uma dessas unidades seja incluída na coluna de soma dessas unidades. Assim, a comparação entre a distribuição das unidades dialógicas no C-ORAL-ESQ e C-ORAL-BRASIL, pelo menos neste momento, vai ocorrer apenas em um nível de soma de todas as unidades dialógicas, e não por cada etiqueta dessas unidades, tal como foi realizado para as unidades textuais.

3.11.6 Disfluências

As disfluências medidas neste trabalho são: enunciados interrompidos, quantidade de retractings, palavras retratadas, duração de pausas preenchidas e duração de pausas silenciosas. No caso dos retractings, também foi possível medir sua quantidade de sílabas e categorizar seu segmento inicial.

Para enunciados interrompidos e retractings, o procedimento para comparar pacientes e não pacientes também foi baseado no subcorpus de comparação, o qual prevê igual quantidade de stanzas com igual quantidade de COBs para ambos os

grupos. Assim, é possível contabilizar quantas stanzas foram interrompidas e com quantos COBs isso ocorreu. Além disso, também são contabilizadas as unidades textuais até a interrupção do enunciado. Consequentemente, é possível ver até que ponto pacientes e não pacientes elaboraram suas stanzas antes da interrupção. O motivo da interrupção não é levado em conta neste trabalho e requisita um viés provavelmente mais individual e qualitativo de pesquisa.

No caso dos retractings, foi elaborado um código que classifica seu segmento inicial em oclusivas, laterais, fricativas, africadas e vogais, separando-as por vozeamento e se o segmento é oral ou nasal. Isso permite verificar possíveis recorrências de classes fonéticas em alguma reformulação e fornecer pistas que possam ajudar a distinguir pacientes com esquizofrenia com pessoas sem essa condição no futuro. Do ponto de vista fonológico, foi possível calcular a quantidade de sílabas dos segmentos por meio da aplicação de regras fonológicas variadas implementadas com a biblioteca de expressões regulares *re,* do Python.

3.11.6.1 Pausas preenchidas

Os procedimentos para extrair pausas preenchidas envolveram novo alinhamento dos áudios utilizados e tratamento em ambiente Python.

No alinhamento tradicional do C-ORAL-BRASIL e C-ORAL-ESQ, apenas enunciados são alinhados. Esse alinhamento produz um arquivo de xml do qual é possível extrair a duração dos trechos alinhados, neste caso, apenas de enunciados.

Nesses áudios, as pausas preenchidas são marcadas pela etiqueta =TMT= (tomada de tempo) em um nível informacional e *&he* em termos de transcrição. Como as pausas preenchidas estão dentro dos enunciados e não alinhadas separadamente, sua duração não pode ser visualizada se são seguidos os critérios tradicionais de alinhamento. Por isso, primeiro foi necessário mapear onde estavam essas pausas e, posteriormente, alinhá-las separadamente.

A localização das pausas foi realizada com mineração de texto. Na Tabela 4 a seguir, é possível ver a ocorrência da pausa na transcrição, em negrito, e o registro de sua frequência na outra coluna. A Tabela 4 fornece a quantidade de pausas preenchidas em cada enunciado dos corpora utilizados em relação à frequência de ocorrência. Para obter a duração dessas pausas, foi necessário utilizar o xml e o áudio fornecido pelos corpora, localizar os enunciados da Tabela 4 e separar as pausas

preenchidas, que são marcadas pela etiqueta =TMT=. Isso gera uma fronteira esquerda e direita na onda acústica, que é refletida no arquivo xml na forma de um tempo inicial e outro tempo final, a partir do qual é possível extrair a duração de cada unidade alinhada. Nesta tese, e também no código de extração previsto para realizar essa operação (cf. Programa 1 no Anexo – A), as pausas alinhadas foram marcadas com a etiqueta =PAUSA_P=.

Tabela 4 – Exemplo de localização de pausas preenchidas nas transcrições.

Áudio	Enunciado	Frequência de pausas preenchidas
bfammn01_2018	**&he** /=TMT= **&he** /=TMT= o	2
bfammn01_2018	**&he** /=TMT= esse rapaz /=TOP= **&he** /=TMT= abriu um [/1]=SCA= um claro dentro de uma mata /=COB= pa fazer uma [/1]=SCA= uma plantação /=COB= fazer &u [/1]=SCA= tipo [/1]=EMP= tipo de lavoura /=COM= né //=AUX=	2

Fonte: Elaborado pelo autor.

Nesse processo, foi considerada toda a extensão do segmento na onda acústica, independentemente de sua duração. Conforme já discutido na subseção 2.5, a duração mínima para uma pausa é bastante controversa, inclusive para pesquisadores da fala de pessoas com esquizofrenia. Neste trabalho, não foi estabelecido um limite mínimo para pausas preenchidas.

Essa etapa da pesquisa visava responder as seguintes questões de pesquisa: as pausas preenchidas dos pacientes têm duração maior ou menor do que a dos não pacientes? Qual foi a frequência observada desse fenômeno?

3.11.6.2 Pausas silenciosas

As pausas silenciosas foram encontradas pelo script de marcação de pausas silenciosas de Mietta Lennes (2006) e tratadas em ambiente Python pelo Programa 4 (Anexo D), feito pelo autor desta tese.

O código de Lennes detecta as pausas na onda acústica por meio de parâmetros acústicos pré-escolhidos no programa Praat (Boersma, 2001). Foram estabelecidos 50 decibéis para intensidade, 100 hertz para frequência fundamental e

200 ms[28] de duração para ser considerado uma pausa silenciosa. Como a maioria dos áudios tinha dois canais, o canal do participante que não seria avaliado – quem não era narrador no C-ORAL-BRASIL e quem não era paciente no C-ORAL-ESQ- foram removidos. Dessa forma, a onda acústica fica mais facilmente visualizável para eventuais modificações nas pausas, que eram marcadas com um “xxx” (padrão do programa de Lennes). Posteriormente, cada áudio foi ouvido integralmente e descartadas eventuais pausas marcadas incorretamente, de acordo com a percepção do autor deste trabalho.

Com o arquivo em formato TextGrid gerado pelo Praat, é possível utilizar o Programa 4. Apesar de o limite de 200 ms já ter sido estabelecido no processo de captura das pausas, não foram incomuns pausas silenciosas marcadas automaticamente pelo programa com duração inferior. Por isso, optou-se por estabelecer como pausa silenciosa as ocorrências que tivessem mais de 180 ms por meio de um filtro no código, conforme explicitado no Anexo D.

3.12 Procedimentos para descrição lexical preliminar do C-ORAL-ESQ

Conforme já explicitado na introdução desta metodologia, esta seção detalha os procedimentos para descrever o léxico e medidas da fala de uma amostra representativa do C-ORAL-ESQ com 16 áudios. Portanto, diferentemente das seções anteriores, que trataram de dois grupos de comparação, esta seção trata apenas de uma descrição de uma amostra mais ampla do C-ORAL-ESQ.

Trata-se de uma descrição preliminar da estrutura dos enunciados e diversas medidas e disfluências da fala que já podem ser obtidas em transcrições que já foram alinhadas, mas não necessariamente etiquetadas informacionalmente. Dessa forma, é possível ter um panorama detalhado, e em quantidade muito maior do que na comparação com o C-ORAL-BRASIL. O objetivo é produzir reflexões que podem ser importantes para a compreensão de disfluências relacionadas à esquizofrenia e, para além disso, oferecer tecnologia em linguagem Python para futuras pesquisas que se debrucem sobre os temas tratados nesta tese ou sob o escopo da Teoria da Língua em Ato.

[28]Limites menores nessa etapa geram resultados mais imprecisos.

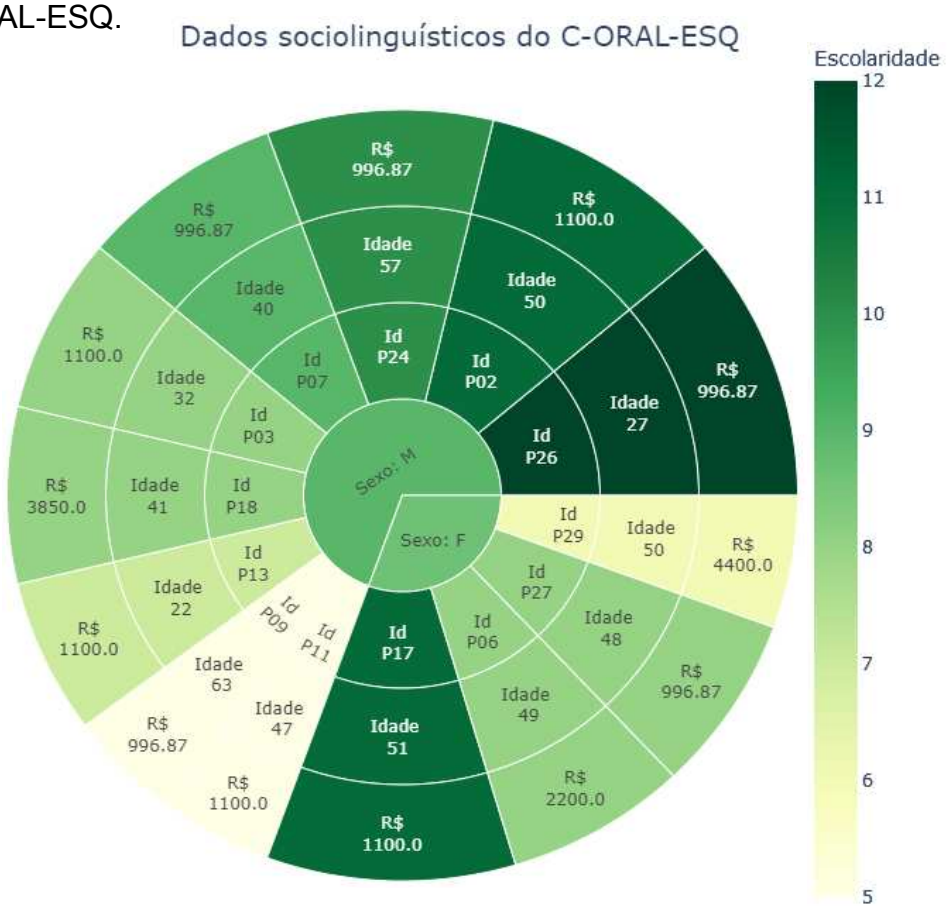
Em relação ao léxico, esta seção se dedica a descrever a normalização ortográfica, etiquetagem por classes gramaticais e detalhamentos diversos que permitiram integrar[29] o DELAF (MUNIZ, 2003; 2004) e a VerboWeb (CANÇADO et al, 2017) para detalhar morfologia e questões sintáticas e semânticas do C-ORAL-ESQ e C-ORAL-BRASIL, principalmente por mineração de texto e web scraping.

Os dados dos pacientes presentes nessas transcrições são descritos na seção 3.13 Já a descrição das medidas da fala é tratada na subseção 3.14, enquanto a discussão sobre os procedimentos para o léxico tem lugar nas seções subsequentes.

3.13 Dados sociolinguísticos e clínicos dos pacientes

Para essa descrição preliminar foram utilizados 16 áudios. Parte desses áudios foi alinhada pelo autor desta tese em fase anterior ao processo de etiquetagem informacional. Já o restante dos áudios foi alinhado por colaboradores do LEEL. Alguns dados sociolinguísticos dos pacientes podem ser visualizados na Figura 7.

Figura 7 – Dados sociolinguísticos dos pacientes da amostra utilizada para descrição do C-ORAL-ESQ.



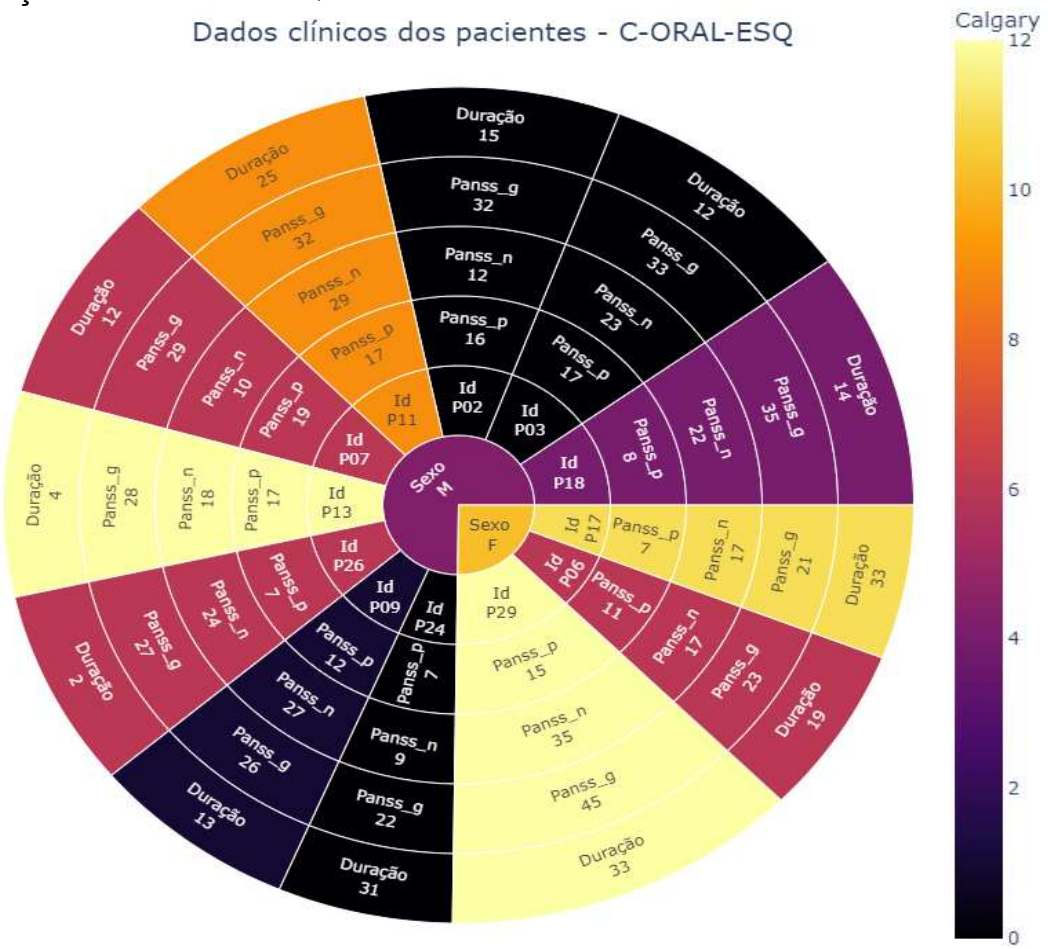
[29]Isso também foi feito para a Framenet Brasil (SALOMÃO, 2021), mas, como ainda está em fase de aperfeiçoamento da implementação, foi disponibilizado apenas no Anexo F.

Fonte: Elaboração do autor.

Há predominância de pacientes do sexo masculino (9) em relação aos do sexo feminino (4) nesta descrição. São 16 áudios de interações distintas, mas 3 pacientes (P09, P024 e P026) participam de duas interações cada. As mulheres têm entre 48 e 50 anos, enquanto os homens têm entre 22 a 63. A renda média desses pacientes é de R$ 996.87, valor utilizado para preencher ocasionais dados faltantes. Observa-se que a maioria dos pacientes analisados é de classe social mais desfavorecida e possuem baixa escolaridade, conforme é possível verificar pela abundância de tons em verde mais claro no gráfico.

Na Figura 8, é possível ver a pontuação desses pacientes nas escalas PANSS de Sintomas Positivos (Panss_p), Negativos (Panss_n), além da PANSS Geral (Panss_g) e da Escala Calgary (termômetro de cor).

Figura 8 – Pontuação em escalas de sintomas dos pacientes da amostra utilizada para descrição do C-ORAL-ESQ.



Fonte: Elaboração do autor.

Há grande diversidade de sintomas de pacientes nesse subcorpus utilizado para descrição preliminar do C-ORAL-ESQ. O escore na escala de PANSS de Sintomas Positivos vai de 7 a 19. Na de Negativos, de 9 a 29, enquanto na PANSS Geral vai de 21 a 45. Na Escala Calgary, que mede o grau de depressão dos pacientes, observa-se desde pacientes pouco deprimidos, marcados com a cor preta, a pacientes muito deprimidos, destacados em cores mais claras.

Os medicamentos prescritos para esses pacientes são explicitados na Tabela 5 a seguir.

Tabela 5 - Prescrição de medicamentos dos pacientes da amostra utilizada para descrição do C-ORAL-ESQ.

Id	Áudio	Prescrição
P02	MED_002_1	Clozapina 400mg/d
P03	MED_003	Quetiapina 200mg/d + Amissulprida 400mg/d + Clonazepam 1mg/d + Nitrazepam 5mg/d
P06	MED_007	Risperidona 3mg/d + Clorpromazina 25mg/d + Fluoxetina 20mg/d + Clonazepam 2mg/d
P29	MED_008	Clozapina 500mg/dia + Haloperidol 10mg/dia
P07	MED_009	Olanzapina 20mg/dia + Diazepam 3mg/dia
P09	MED_011	Risperidona 6mg/dia + Quetiapina 50mg/dia + Levomepromazina 25mg/dia
P11	MED_013	Clozapina 600mg/dia + Haloperidol 5mg/dia + Prometazina 100mg/dia + Clorpromazina75mg/dia + AVP 750mg/dia + Biperideno 2mg/dia
P13	MED_015	Olanzapina 10mg/dia
P17	MED_019	Olanzapina 25mg/dia + AVP 500mg/d + Fluoxetina 20mg/d + Clorpromazina 100mg/d
P18	MED_020	Quetiapina 200mg/dia
P24	MED_027_2	Trifluoperazina 10mg/dia + AVP 1500mg/dia
P26	MED_029_2	Risperidona 2,5mg/dia
P27	MED_031	Olanzapina 30mg/dia + AVP 1500mg/dia

Fonte: Elaboração do autor.

3.14 Descrição da estrutura dos enunciados e medidas da fala do C-ORAL-ESQ

Em grande medida, as questões e variáveis discutidas nesta seção requisitaram os mesmos procedimentos descritos para realizar a comparação entre os grupos do C-ORAL-ESQ e do C-ORAL-BRASIL, descritos com pormenores nas seções 3.11. Além de contar apenas com as transcrições dos áudios dos pacientes, outra diferença importante a ser destacada é que, neste caso, não se trata de áudios etiquetados informacionalmente, isto é, não há etiquetas de unidades informacionais que permitam identificar o padrão informacional desses pacientes. Apesar disso, o processo para extração dessas informações é o mesmo, isto é, basta utilizar o

Programa 1 (Anexo A). Por isso, o foco das seções subsequentes será o léxico e os procedimentos requisitados para tratá-lo, pois isso ainda não foi explicitado.

Dito isso, as variáveis a serem contempladas na descrição dos enunciados e de medidas da fala são:

I. Estrutura geral dos enunciados: Visão esquemática do corpus em um único gráfico de frequência de categorias, a partir do qual é possível visualizar e quantificar a frequência de unidades tonais, duração, número de palavras, palavras por segundo, retractings, palavras retratadas, pausas preenchidas, enunciados interrompidos e número de sobreposições
II. Correlação de Spearman entre variáveis.
III. Quantidade de palavras e duração.
IV. Enunciados simples e complexos.
V. Unidades tonais.
VI. Estrutura geral de enunciados interrompidos antes de sua interrupção.
VII. Retractings e palavras retratadas: frequência de ocorrência e por quantidade de sílabas, classificação do segmento inicial de acordo com sua classe fonética.
VIII. Pausas preenchidas: frequência e duração.
IX. Pausas preenchidas diante de classes de palavras: frequência por classe e duração.

Para realizar essas operações até VII, é necessário rodar o Programa 1 (Anexo A) com as transcrições em xml tradicionais, já alinhadas e fornecidas pelo LEEL. Para realizar os procedimentos VIII e IX, é necessário alinhar as pausas preenchidas separadamente e marcar com =PAUSA_P= o trecho alinhado. A partir disso, é possível rodar também no Programa 1 e obter essas medições automaticamente.

3.15 Normalização ortográfica

A normalização, incorporada ao Programa 1, é feita por meio de uso extensivo da biblioteca re, de expressões regulares, e filtragens diversas pelo Pandas. Trata-se de um procedimento bastante complexo que transforma a transcrição semiortográfica do C-ORAL-ESQ e C-ORAL-BRASIL em ortografia padrão do português automaticamente. Esse processo viabiliza etiquetagem de classes de palavras e quaisquer outras operações que demandem ortografia padrão.

Nesse processo, são eliminadas as reformulações, chamadas de retractings, todas as etiquetas de disfluência, como enunciados interrompidos (+), pausas preenchidas (&he), ruídos paralinguísticos (hhh, para risadas, xxx e yyy para palavras e trechos ininteligíveis) entre muitos outros critérios de transcrição (ver MELLO, RASO et al, 2012, pp.125-145 para detalhes).

Após essa limpeza, o programa utiliza um microcorpus de cabeçalhos armazenado em seu interior para transformar formas não padrão em formas padrão. Neste caso, trata-se principalmente de aféreses (bacaxi > abacaxi, trapalhou > atrapalhou) e de formas convencionalizadas pela transcrição (pr' ocês > para vocês; nũ > não; siora > senhora, entre muitas outras). O quadro a seguir traz exemplos desse processo.

Quadro 2 – Exemplo de normalização automática da ortografia.

Enunciados	Enunciados Normalizados
ela me fez uma compra lá /=COB_r= comprou presente pr' ocês todo /=COB_r= pos menino /=COB_r= roupa /=COB_r= e tal /=COB_r= e /=DCT_r= ea nũ me pagou /=CMM_r= também nũ cobrei /=CMM_r= e já perdoei isso há muito tempo /=CMM_r= nem converso mais não //=COM_r=	ela me fez uma compra lá comprou presente para vocês todo para os menino roupa e tal e ela não me pagou também não cobrei e já perdoei isso há muito tempo nem converso mais não
mas ô mãe /=CMM_r= mas nũ fica bem /=CMM_r= mãe //=CNT_r=	mas ô mãe mas não fica bem mãe
siora despede lá /=COB_r= e /=SCA_r= sai o enterro //=COM_r=	senhora despede lá e sai o enterro
eu te [/1]=SCA_r= te avisei /=CMM_r= nũ foi //=CMM_r=	eu te te avisei não foi
cê nũ me escutou //=COM_r=	você não me escutou
<nũ tem nada errado aí não> //=COM_r=	não tem nada errado aí não

Fonte: Elaboração do autor.

Possíveis inadequações de concordância nominal e verbal são mantidas. Por exemplo: *Os menino* continua *os menino*. Já *es vai* se transforma em *eles vai* e não *eles vão*. Isso ocorre por dois motivos. Primeiramente, para não corrigir a forma como o participante se expressa na fala tendo por base convenções normativas, sejam elas escritas ou faladas. O outro motivo é que é o desafio de normalização automática se torna de dificuldade muito maior, visto que também há a forma verbal *vão*, assim como há o substantivo *meninos*, e nem todo *menino* e tampouco *vai* precisam ser normalizados para *meninos* e *vão*, respectivamente.

A partir disso, é possível partir para a etiquetagem de classes gramaticais, explicitada na seção a seguir.

3.16 Etiquetagem de classes de palavras

A etiquetagem é feita após o treinamento de um etiquetador do tipo Brill no Mac-Morpho (ALUÍSIO et al, 2003), corpus gratuito, disponível na NLTK, com 51.397 sentenças já etiquetadas, o que totaliza 1.170.095 palavras. Parte desse código é inspirado no de Mateus Inoue[30] e na ampla documentação da NLTK[31]desse processo, na qual o código de Inoue também foi baseada. A diferença principal é no tratamento extensivo com o Pandas após a etiquetagem, de acordo com as necessidades das transcrições do C-ORAL-ESQ e C-ORAL-BRASIL, pois, apesar de o Mac-Morpho ser um corpus bastante representativo do português escrito, isso não se aplica integralmente à fala espontânea.

Os etiquetadores utilizados são o DefaultTagger[32], o AffixTagger; UnigramTagger, BigramTagger, TrigramTagger e BrillTagger. Apesar de poder ser utilizado o RegexTagger, de expressões regulares, como backoff dos outros, a eficácia correção da etiquetagem por meio do módulo re no Pandas se mostrou muito mais eficiente. Por isso, não é recomendado utilizar esse etiquetador de classes de palavras fora do conjunto de códigos do Programa 1, uma vez que é o Pandas e a mineração de texto, presentes no programa e não no etiquetador, que fazem diversas correções posteriores ao processo que envolve a NLTK aqui descrito.

Cada um desses etiquetadores tem a função de fornecer uma base para o etiquetador anterior e melhorar sua acurácia, a partir de cálculos de probabilidade que envolvem procedimentos diferentes. Primeiramente, é atribuído um etiquetador default, chamado de DefaultTagger. Esse etiquetador recebe a etiqueta provavelmente mais frequente no corpus que, no caso do Mac-Morpho, é o substantivo (N). Para os C-ORAL-ESQ e C-ORAL-BRASIL, também foi utilizado esse procedimento e, quando todos os etiquetadores que o sucedem falham em reconhecer a classe de palavra em questão, a esta é atribuído um “N”, de substantivo, pois a probabilidade de ser um substantivo é maior do que todas as outras. A acurácia desse etiquetador é de apenas 19% de acerto.

[30]https://github.com/inoueMashuu
[31]https://www.nltk.org/book/ch05.html

Posteriormente, foi construído um AffixTagger, que foi programado para verificar a periferia direita das palavras, uma vez que, em português, o sufixo pode ajudar na identificação de uma classe de palavra. De fato, a acurácia do modelo aumenta, mas ainda é bastante baixo, com 36% de acerto.

Após esse etiquetador, foi treinado um UnigramTagger, que basicamente é o sentido dicionarizado da palavra. Há um salto de acurácia bastante importante, para 83,75% de acerto. A partir deste ponto, os ganhos obtidos com o BigramTagger, que combina duas palavras, e com o TrigramTagger, que combina três, são mínimos, com 85,22% e 85,24%, respectivamente. É com o BrillTagger, que calcula regras funcionais e úteis a etiquetagem baseado no corpus de treinamento, que há um salto final de acurácia, com 92,24% de acerto. A seguir, exemplos dessa etiquetagem escolhidos aleatoriamente.

Quadro 3 – Exemplo de enunciados normalizados etiquetados por classes de palavras.

Enunciados etiquetados por classes de palavras
[('porque', 'KS'), ('minha', 'PROADJ'), ('mulher', 'N'), ('queria', 'V'), ('que', 'KS'), ('eu', 'PROPESS'), ('queimasse', 'V'), ('isso', 'PROSUB'), ('há', 'V'), ('muitos', 'PROADJ'), ('ano', 'N'), ('porque', 'KS'), ('esses', 'PROADJ')]
[('ele', 'PROPESS'), ('veio', 'V'), ('agora', 'ADV'), ('tem', 'V'), ('uns', 'ART'), ('três', 'NUM'), ('anos', 'N')]

Fonte: Elaborado pelo autor

3.17 Detalhamento do léxico

Essa etapa da pesquisa visava a fornecer informações variadas sobre o léxico e classes gramaticais encontradas no C-ORAL-ESQ. O objetivo é possibilitar maior detalhamento do léxico de pessoas com esquizofrenia e aumentar a qualidade da descrição de forma automatizada. As informações contempladas nesse processo são:

I. Lematização (para classes variáveis) – com DELAF e Spacy.
II. Estemização da palavra com o RSLP Stemmer, da NLTK.
III. Detalhes morfológicos como flexões de verbos e nomes e classificação verbal bastante precisa a partir das informações do DELAF.
IV. Contagem da frequência de todos os tempos e modos verbais da transcrição em questão, assim como das flexões de substantivos e adjetivos;

V. Contagem de sílabas por meio da aplicação de regras fonológicas variadas inseridas no código com esta função.
VI. Classificação fonética do segmento inicial de classes gramaticais.
VII. Gráficos variados e automáticos, como nuvem de palavras customizada e frequência de classes gramaticais.
VIII. Classificação do aspecto lexical, papel temático, estrutura sintática, transitividade e classe verbal de acordo com a VerboWeb, a partir de dados obtidos por web scraping[33], de acordo com o lema obtido no C-ORAL-ESQ.

Para I, III e IV, foi criada uma conexão entre o DELAF e as transcrições do C-ORAL-ESQ e do C-ORAL-BRASIL que são utilizadas no Programa 2 (cf. Anexo B). Esse procedimento é descrito na seção a seguir, enquanto o web scraping e incorporação dos dados da VerboWeb são descritos na seção subsequente.

3.17.1 Lematização e detalhes morfológicos com o DELAF

O DELAF utilizado[34] possui uma série de informações a respeito de palavras de classes lexicais, a saber, substantivos, adjetivos, verbos e advérbios. São 878.654 entradas que fornecem uma entrada, que pode ser flexionada, o lema dessa entrada e informações morfológicas, de acordo com sua classe gramatical. Isso pode ser visto no exemplo a seguir.

Exemplo 2:

digeriu,digerir.V:J3s

Nesse exemplo, a entrada é a forma verbal *digeriu*. A seguir, está o lema da palavra, que representa sua forma – verbal, neste caso – mais básica, que é o infinitivo. Depois, a classe à qual a entrada pertence, que é o Verbo (V). Posteriormente, há as informações do Tempo e Modo, que, neste caso, é o Pretérito Perfeito do Indicativo. A seguir, estão as flexões de pessoa e número, que são 3ª pessoa e singular, respectivamente.

A partir de operações de mineração de texto e limpeza de dados, foi possível separar as classes previamente etiquetadas (cf. seção 3.16) e comparar sua forma com as do DELAF. Se são iguais, o Programa 2 vai requisitar as informações

[34] http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html

disponíveis do DELAF, de acordo com a classe gramatical em questão (cf. Anexo B para maiores detalhes do Programa 2 e acesso ao código[35]).

Nesta tese, esse detalhamento é feito com as classes dos substantivos, adjetivos e verbos mais lexicais, isto é, verbos auxiliares não são detalhados nesse momento. Todos os arquivos gerados pelo Programa 2, inclusive o arquivo de *input* utilizado, com os 16 áudios da descrição preliminar do capítulo IV, podem ser consultados[36].

Na Tabela 6 a seguir, gerada de forma aleatória, há dez exemplos de como esse detalhamento é feito com o DELAF, neste caso, com verbos.

Tabela 6 – Exemplo de detalhamento morfológico de verbos

verbos	lema	raiz_nltk	sílabas	classificacao_verbal_delaf
doeu	doer	doeu	2	['Pretérito Perfeito do Indicativo', '3s']
colocou	colocar	coloc	3	['Pretérito Perfeito do Indicativo', '3s']
melhorou	melhorar	melhor	3	['Pretérito Perfeito do Indicativo', '3s']
constatando	constatar	constat	4	['Gerúndio']
morde	morder	mord	2	['Presente do Indicativo', '3s', 'Imperativo', '2s']
andar	andar	and	2	['Infinitivo', '1s', 'Infinitivo', '3s', 'Futuro do Subjuntivo', '1s', 'Futuro do Subjuntivo', '3s']
plantando	plantar	plant	3	['Gerúndio']
reconhece	reconhecer	reconhec	4	['Presente do Indicativo', '3s', 'Imperativo', '2s']
cobre	cobrir	cobr	2	['Presente do Indicativo', '3s', 'Imperativo', '2s']
cortei	cortar	cort	2	['Pretérito Perfeito do Indicativo', '1s']

Fonte: Elaborado pelo autor.

Essa mesma amostra é utilizada na seção subsequente para exemplificar outros detalhes dos verbos dos pacientes, os quais são discutidos a seguir.

3.17.2 Detalhamento de verbos com web scraping e mineração de texto

Além das informações morfológicas já especificadas na seção anterior, foi realizado web scraping[37] na VerboWeb para complementar a descrição dos verbos

[35]Vídeo instrutivo de como é feita a extração com o DELAF:
https://www.loom.com/share/d506efde2b7e4f268f777a07976f7d3b
Resultados: https://www.loom.com/share/f8777b37592b436d8687b00f85fbdd6c

[36]https://drive.google.com/drive/folders/1oR0e-FcAfl42UlGUNF4cfKL_WD2ssNhb?usp=sharing
[37]Vídeo ilustrativo: https://www.loom.com/share/8b6277d286be456b85b6af28354ad945

utilizados pelos pacientes. Para isso, foi construído um robô (cf. Anexo E) que entra nesse site e coleta as seguintes informações (cf. CANÇADO, AMARAL, 2016, para detalhes):

I. Classe do verbo, isto é, se é de Causação, Atividade, Culminação ou Estado.
II. Aspecto lexical, a saber, se o verbo é de *achievement, accomplishment*, atividade, estado ou estado complexo.
III. Papéis temáticos, que são Agente, Causa, Estímulo, Paciente ou Experienciador, Locativo, Alvo e Tema.
IV. Estrutura sintática requisitada pelo verbo, por exemplo [SN V SN].
V. Transitividade, isto é, se o verbo é transitivo, bitransitivo, intransitivo ou inacusativo.

A Tabela 7 a seguir ilustra esse procedimento e utiliza os mesmos verbos da Tabela 6.

Tabela 7 – Exemplo de informações obtidas por web scraping na VerboWeb.

verbos	classe_verbo_web	aspecto_lexical	papel_tematico	transitividade
doeu	Atividade: verbos internamente causados (inergativos)	atividade	Agente	intransitivo
colocou	Causação: verbos mudança de estado locativo	accomplishment	Agente, Paciente, Locativo	bitransitivo
melhorou	Culminação: verbos de mudança de estado	achievement	Paciente	intransitivo
constatando	Culminação: verbos de mudança de estado mental	achievement	Paciente/Experienciador, Objeto Estativo	transitivo
morde	Atividade: verbos de contato mediado pelo corpo	atividade	Agente, Objeto Afetado	transitivo
andar	Atividade: verbos internamente causados (inergativos)	atividade	Agente	intransitivo
plantando	Causação: verbos mudança de estado locativo	accomplishment	Agente, Paciente, Locativo	bitransitivo
reconhece	Culminação: verbos de mudança de estado mental	achievement	Paciente/Experienciador, Objeto Estativo	transitivo
cobre	Causação: verbos de mudança de estado de posse	accomplishment	Causa ou Agente, Paciente, Instrumento	bitransitivo
cortei	Atividade: verbos de contato mediado por instrumento	atividade	Agente, Objeto Afetado	transitivo

Fonte: Elaborado pelo autor.

Todas essas informações são incorporadas ao C-ORAL-ESQ a partir do lema verbal, também disponível na VerboWeb. Assim, se o lema verbal do C-ORAL-ESQ é igual, todas as informações elencadas de I a V nesta seção também estarão disponíveis e enriquecendo a descrição dessa classe lexical dos pacientes.

4 DESCRIÇÃO PRELIMINAR DO C-ORAL-ESQ – MEDIDAS DA FALA E LÉXICO

Este capítulo compreende a descrição preliminar do C-ORAL-ESQ em relação à estrutura de seus enunciados, tanto do ponto de vista de das medidas da fala quanto de seu léxico. Questões mais estruturais de organização e frequência de enunciados, quantidade de palavras e duração, além de ocorrência de enunciados simples e complexos são discutidas das seções 4.1 a 4.3. As disfluências registradas na fala dos pacientes ocupam das seções 4.4 a 4.6, a saber, enunciados interrompidos, retractings e pausas preenchidas – com suas possíveis relações com classes gramaticais - respectivamente. As seções e subseções subsequentes tratam mais especificamente da distribuição das palavras mais frequentes do subcorpus utilizado, além de características morfológicas ou sintáticas de algumas classes de palavras utilizadas para ilustrar essa descrição, neste caso, substantivos, adjetivos e verbos.

4.1 Quantidade de enunciados por áudio

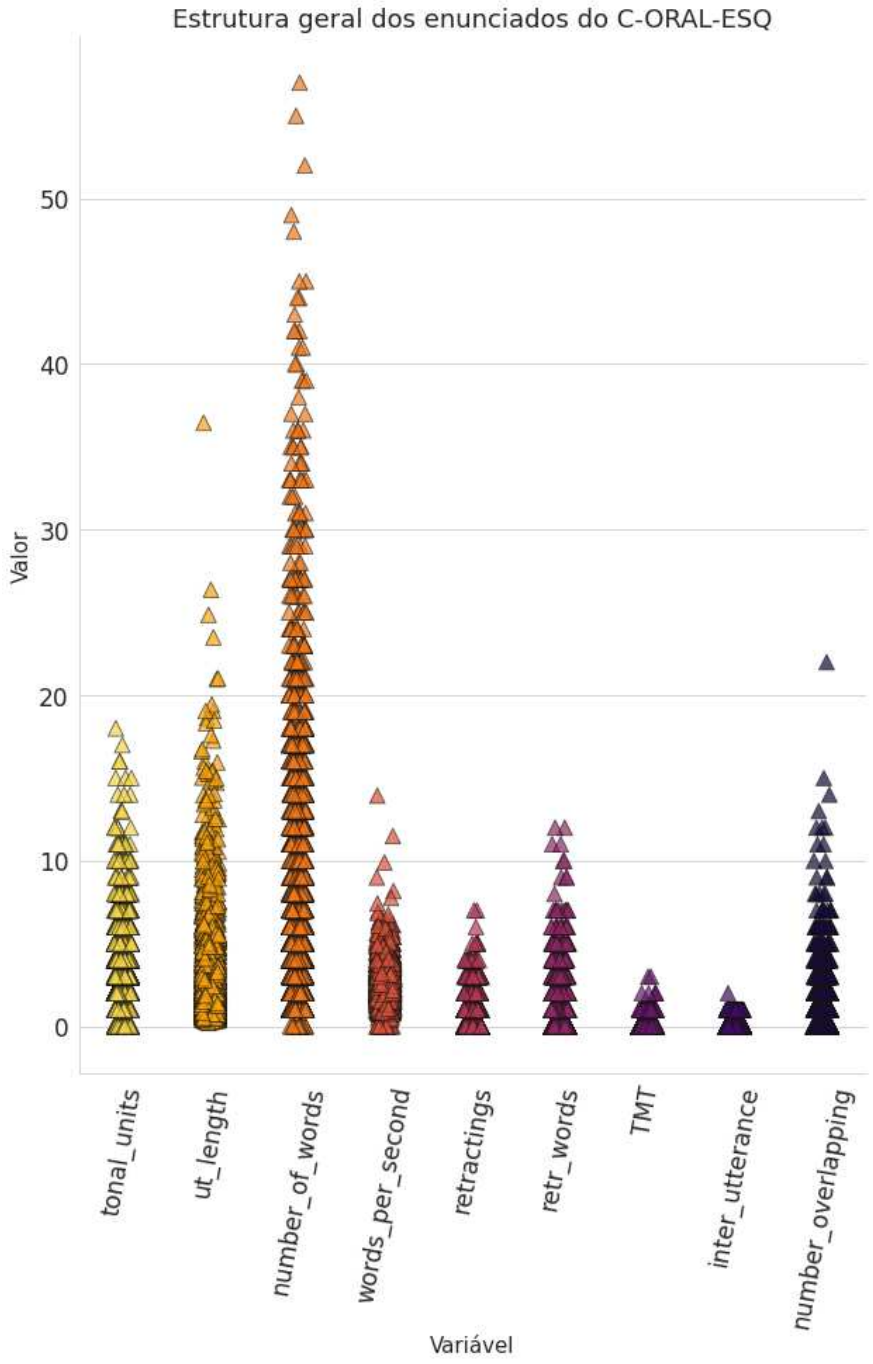
O subcorpus montado com os 16 áudios do C-ORAL-ESQ possui 3.169 enunciados. Esses enunciados se distribuem de forma bastante heterogênea entre os áudios (média = 198,06; std = 91,30). Há consultas em que alguns pacientes falam de fato muito pouco, como em MED_008, MED_011 e MED_029_1.

A composição interna desses enunciados pode ser esquematizada na Figura 9. Trata-se de uma visão esquemática das principais variáveis discutidas e detalhadas a seguir. Nessa visualização, é possível visualizar a distribuição de unidades tonais, duração, quantidade de palavras, palavras por segundo, retractings, palavras retratadas, pausas preenchidas, enunciados interrompidos e quantidade de palavras que o paciente sobrepôs à fala do médico durante a consulta. Essas variáveis são descritas em linhas gerais a seguir e de forma detalhada nas seções subsequentes.

A distribuição das unidades tonais se concentra abaixo de 10 e vai até cerca de 18, enquanto a duração dos enunciados do subcorpus se concentra abaixo de 15 segundos e atinge picos em torno de 35 segundos. Observa-se que prevalecem enunciados abaixo de 30 palavras, com ocasionais picos que atingem até acima de 50. Por sua vez, a quantidade de palavras por segundo se concentra em torno no

intervalo de 0 a 5, com escassas ocorrências até 13. Em relação aos retractings, é possível verificar que estes se concentram em até 2 ou 3 nos enunciados, com outliers que atingem em torno de 5, ao passo que as palavras retratadas se concentram em torno de 3 e chegam até acima de 10.

Figura 9- Esquematização da estrutura dos enunciados da amostra do C-ORAL-ESQ.

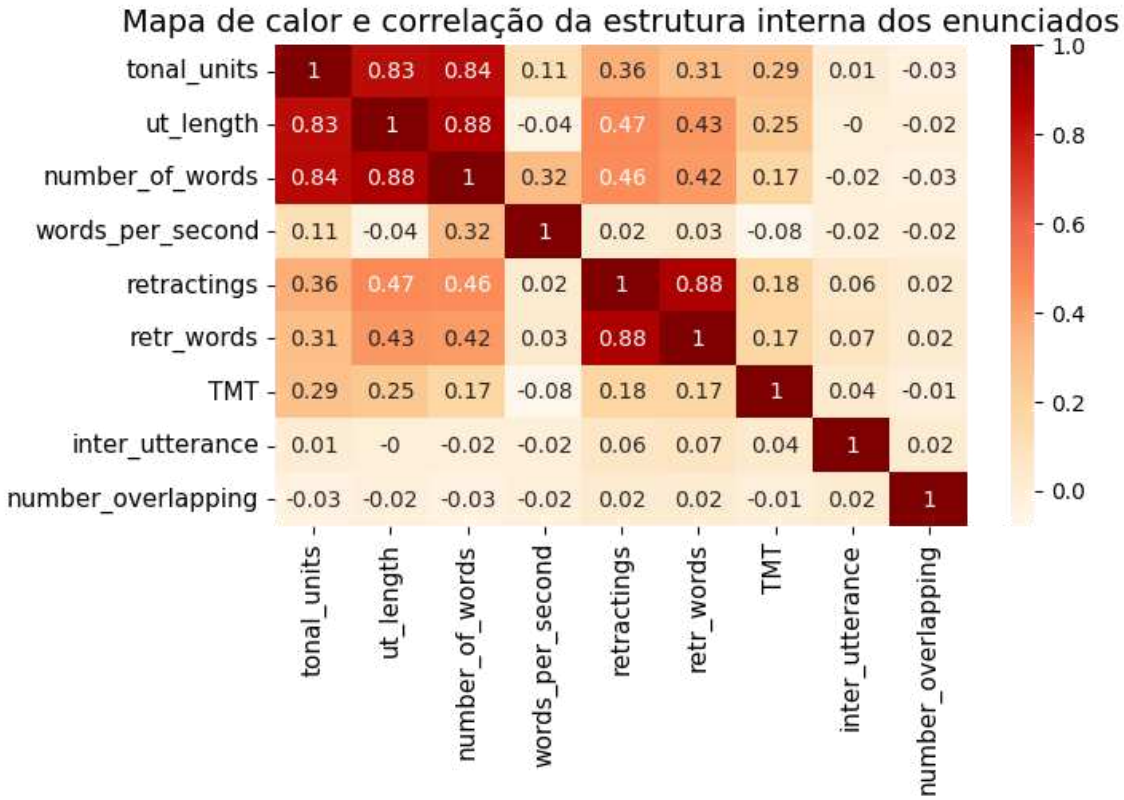


Fonte: Elaboração do autor.

No que concerne às pausas preenchidas (TMT) e enunciados interrompidos, ambas parecem se concentrar em torno de 0 ou 1, com ocasionais 2 ocorrências para a primeira. É esperado que os enunciados interrompidos marquem essa frequência, uma vez que o alinhamento é passado ao enunciado subsequente quando de sua ocorrência, de forma que cada enunciado não possui mais de uma ocorrência de enunciado interrompido. Por fim, observa-se que os pacientes sobrepuseram até quase 20 palavras à fala de seus médicos, mas esta distribuição se concentra até em torno de 5 palavras sobrepostas.

Após plotadas em um mapa de calor (Figura 10), que marca a correlação, é possível observar que esse índice é forte (acima de 0,7) apenas entre variáveis que naturalmente influenciam outras, como a quantidade de unidades tonais e tamanho do enunciado ou entre retractings e palavras retratadas, por exemplo. Os valores apresentados entre sobreposições dos pacientes ou enunciados interrompidos e outras variáveis, que poderiam indicar o quanto outras variáveis poderiam explicar a ocorrência desses fenômenos, são desprezíveis e fortemente randômicos.

Figura 10 – Correlação de Spearman entre variáveis da estrutura do enunciado e disfluências gerais

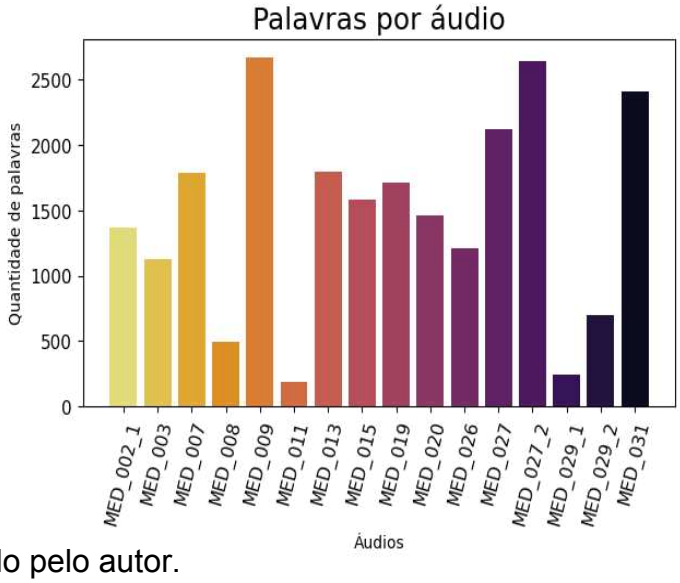


Fonte: Elaborado pelo autor.

4.2 Quantidade de palavras e duração

O subcorpus utilizado para descrição possui 23.526 palavras, com média de 1.470 palavras. A quantidade de palavras por áudio é diversificada, conforme pode ser visto na Figura 11.

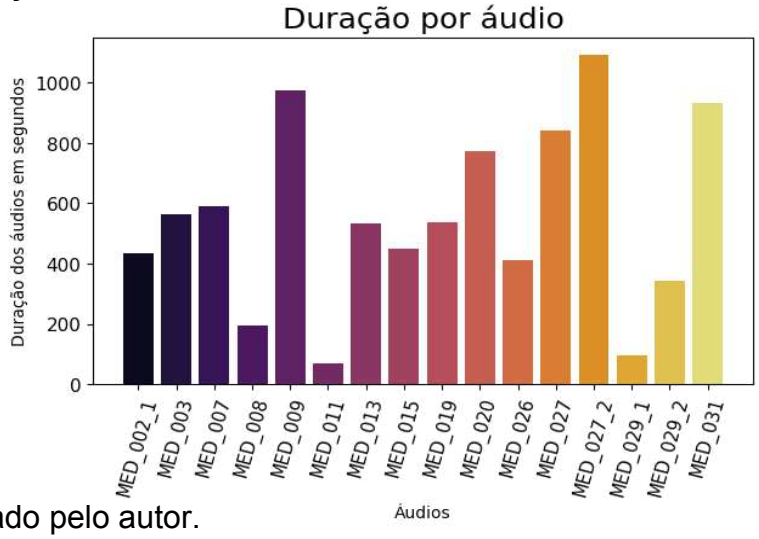
Figura 11 – Palavras por áudio no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Há 4 áudios particularmente com poucas palavras, como MED_008, MED_011, MED_029_1 e MED_029_2, abaixo de 500, e 2 áudios acima de 2500 palavras. De forma geral, a duração também acompanha os contornos da distribuição da quantidade de palavras, isto é, os mesmos áudios que apresentaram picos de quantidade de palavras são os mais longos. Isso pode ser visto na Figura 12.

Figura 12- Duração dos áudios no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

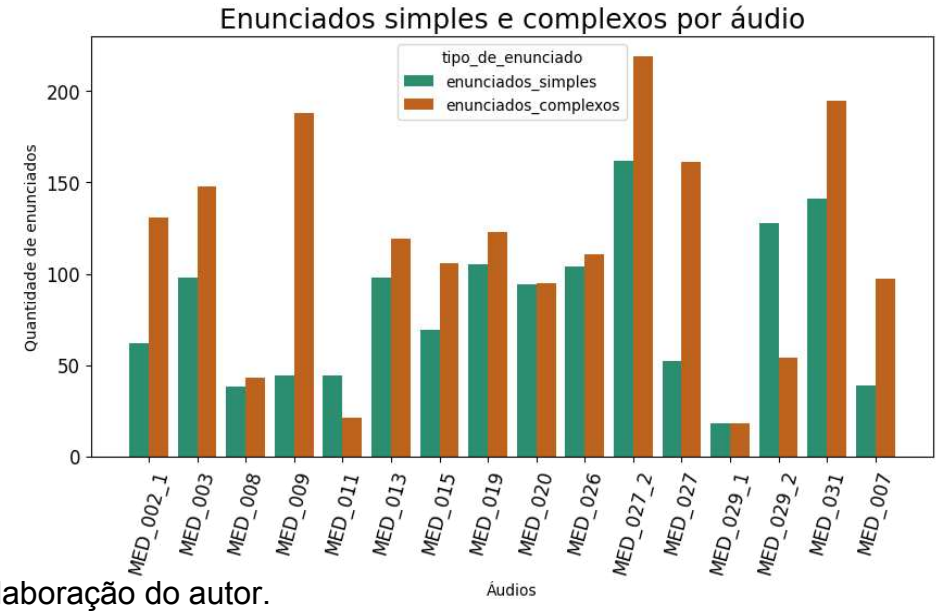
De forma geral, a duração também acompanha os contornos da distribuição da quantidade de palavras, isto é, os mesmos áudios que apresentaram picos de quantidade de palavras são os mais longos. De forma análoga, os áudios com poucas palavras são também os mais curtos. Apesar da afirmação aparentemente evidente, é necessário destacar que alguns pacientes, como o do áudio MED_020, apresenta menor quantidade de palavras do que vários outros, mas duração notavelmente superior. Analisados em conjunto, este é um fator que ajuda a identificar pacientes que demoram, por alguma razão, a realizar seus enunciados.

De forma análoga, os áudios com poucas palavras são também os mais curtos. Apesar da afirmação aparentemente evidente, é necessário destacar que alguns pacientes, como o do áudio MED_020, apresenta menor quantidade de palavras do que vários outros, mas duração notavelmente superior. Analisados em conjunto, este é um fator que ajuda a identificar pacientes que demoram, por alguma razão, a realizar seus enunciados.

4.3 Enunciados simples e complexos

A Figura 13 a seguir apresenta a frequência de ocorrência de enunciados simples e complexos do C-ORAL-ESQ.

Figura 13 – Enunciados simples e complexos no C-ORAL-ESQ

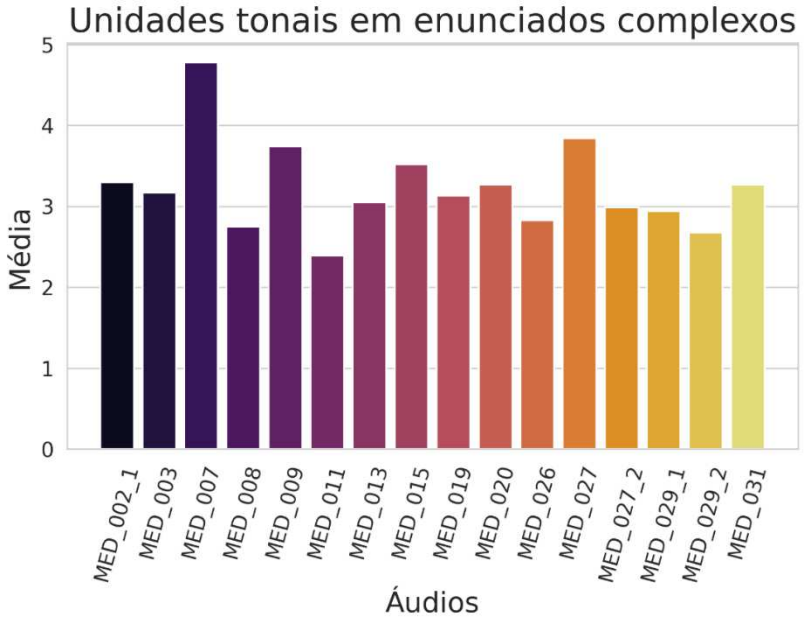


Fonte: Elaboração do autor.

Há um predomínio de enunciados complexos sobre os simples nos áudios da maioria dos pacientes. Enunciados simples são, em geral, frequentes no início das interações, principalmente após perguntas de rotina sobre a saúde do paciente. É esperado, portanto, que haja menos enunciados simples do que complexos, visto que o paciente tende a realizar enunciados maiores quando fala de suas vivências. Quando isso não acontece, isto é, há quantidades semelhantes de enunciados simples e compostos na consulta psiquiátrica, como em MED_008, MED_020, MED_026 e MED_029_2, pode ser um sinal de que o paciente desenvolva pouco seus enunciados, o que poderia indicar, em um primeiro momento, padrões informacionais mais simplificados.

Esses enunciados complexos têm, em média, 3,22 unidades tonais (std = 0,56), com até quase 5 unidades tonais por enunciado no áudio MED_007, única interação que de fato é bastante destoante das demais neste sentido, conforme pode ser visto na Figura 14.

Figura 14 – Unidades tonais em enunciados complexos do C-ORAL-ESQ



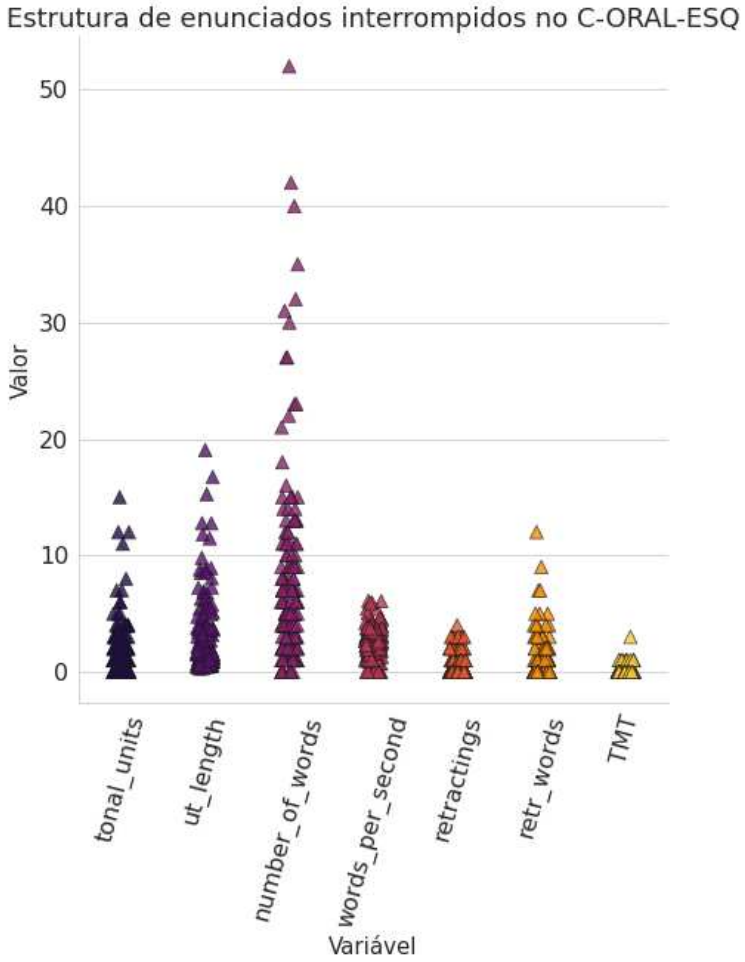
Fonte: Elaboração do autor.

4.4. Enunciados interrompidos

No total, são 193 enunciados interrompidos no C-ORAL-ESQ. Na Figura 15, é possível observar que enunciados desta categoria foram realizados com entre 0 e 15

unidades tonais (média = 1,37; std = 2,15). Tais enunciados concentram sua duração entre 0 e 20 segundos (média = 2,72 segundos; std = 3,03) e possuem em torno de 12 palavras (média = 6,90; std = 7,95), com picos ocasionais acima de 20 palavras. Além disso, observa-se que a quantidade de palavras por segundo é menor do que 5 (média = 2,60; std = 1, 24). Também é possível notar que retractings se concentram entre 0-5 (soma total = 85; média = 0,44; std = 0,81), algo também válido para as palavras retratadas, com picos próximos a 10 para esta última variável (média = 0,70; std = 1,58). Por fim, constata-se que as pausas preenchidas estão entre 0 e 2 (média de 0,06; std = 0,30).

Figura 15 – Estrutura de enunciados interrompidos do C-ORAL-ESQ.

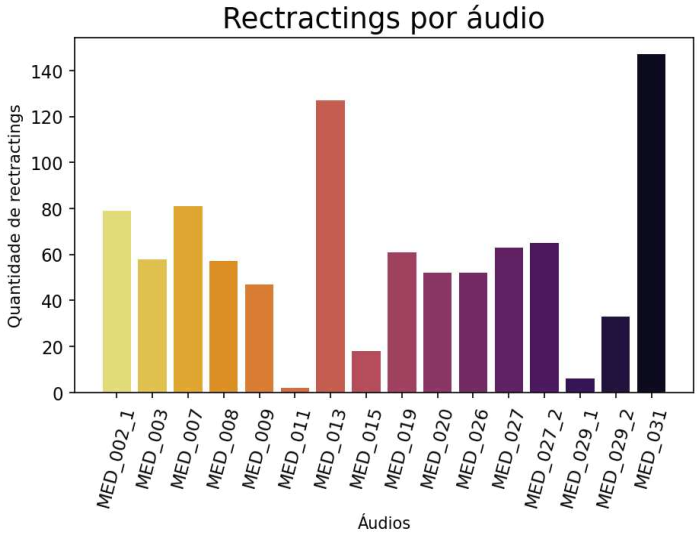


Fonte: Elaborado pelo autor.

4.5 Retractings

A descrição dos retractings registrados considera sua distribuição e o segmento fonético pelo qual se iniciam, além de sua dimensão silábica. Na Figura 16, é possível observar a frequência desse fenômeno por áudio.

Figura 16 – Retractings por áudio no C-ORAL-ESQ

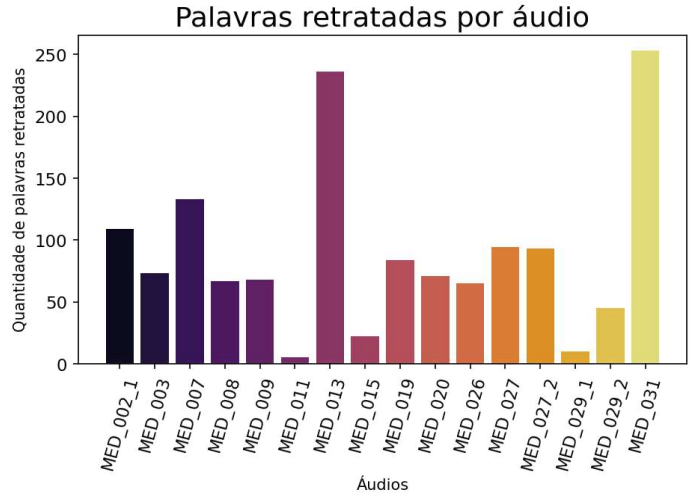


Fonte: Elaboração do autor.

No total, são 948 retractings no C-ORAL-ESQ e média de 59,25 (std = 38,27) por áudio. Essa grande dispersão é alçada pelos áudios MED_013 e MED_031, os quais correspondem a dois pacientes que lideram esse tipo de disfluência, bem como é abaixada pelos áudios curtos MED_011 e MED_029_1, conforme é possível ver na Figura 16. A mediana, entretanto, se difere pouco da média e registra 57,5 retractings por áudio.

Em média, esses retractings apresentaram 89,25 palavras retratadas (std = 69,62, Figura 17) por áudio, palavras essas que se distribuem de forma heterogênea em cada transcrição.

Figura 17 – Palavras retratadas por áudio no C-ORAL-ESQ.



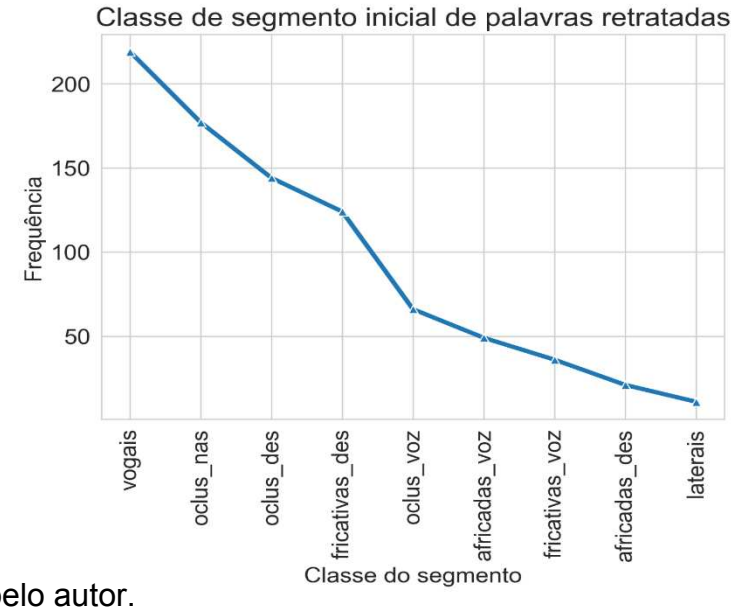
Fonte: Elaboração do autor.

É possível destacar que áudios que apresentaram grande quantidade desse fenômeno também foram os mesmos que registraram a maior quantidade de palavras

retratadas. É preciso destacar que esta análise não leva em conta o motivo pelo qual essa reformulação acontece.

Se agrupados por classe do segmento pelo qual se inicia a palavra retratada, observa-se o predomínio de vogais, oclusivas nasais, oclusivas desvozeadas e fricativas desvozeadas, com uma queda brusca desta última para as classes subsequentes, conforme é possível ver na Figura 18.

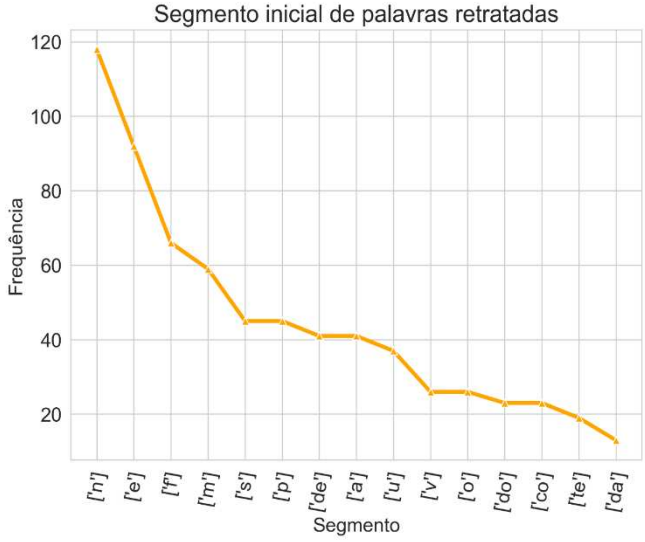
Figura 18 – Classe fonética do segmento inicial de palavras retratadas no C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

Nos segmentos iniciais mais frequentes (Figura 19) das palavras retratadas, observa-se um predomínio bastante destoante da oclusiva nasal /n/. Segmentos que representam oclusivas orais, como /d/ ([do], [da]), /k/ ([co]) e africados desvozeados /tch/ ([te]) foram pouco frequentes.

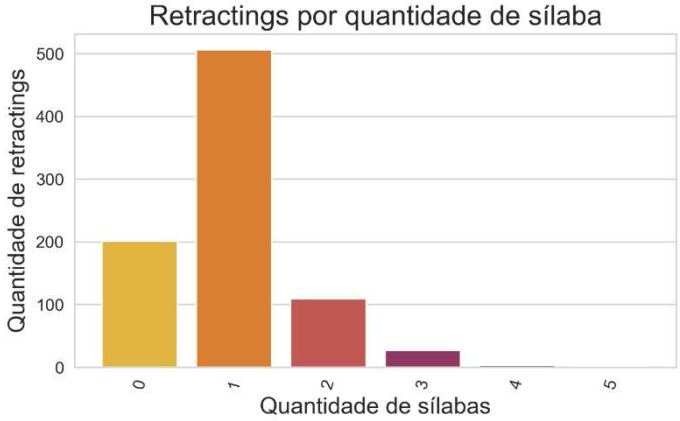
Figura 19 – Segmento inicial de palavras retratadas



Fonte: Elaborado pelo autor.

Essas reformulações têm, em sua maioria, 1 ou 0 sílabas, conforme é possível ver na Figura 20.

Figura 20 – Quantidade de sílabas dos retractings no C-ORAL-ESQ



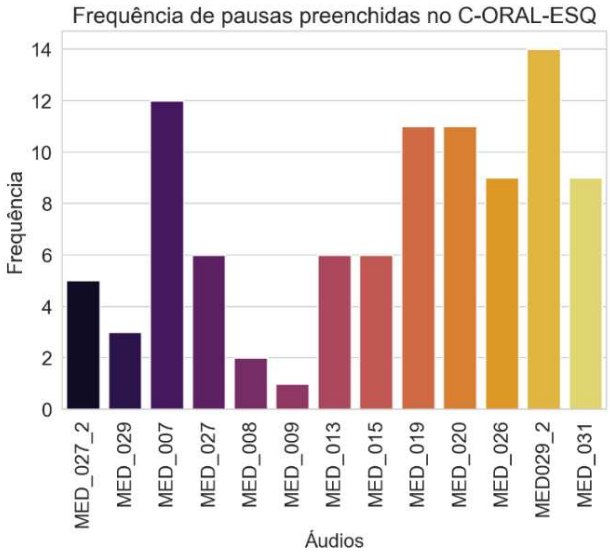
Fonte: Elaborado pelo autor.

Maior quantidade de sílabas por retractings poderia ser um indicativo de que o paciente demora mais a perceber que errou em sua reformulação. As ocorrências de palavras com 5 ou 4 sílabas são desprezíveis.

4.6 Pausas preenchidas

No total, há 94 pausas preenchidas no C-ORAL-ESQ[38] (Figura 21), as quais se distribuem de forma bastante heterogênea nos áudios da amostra.

Figura 21- Frequência de pausas preenchidas no C-ORAL-ESQ



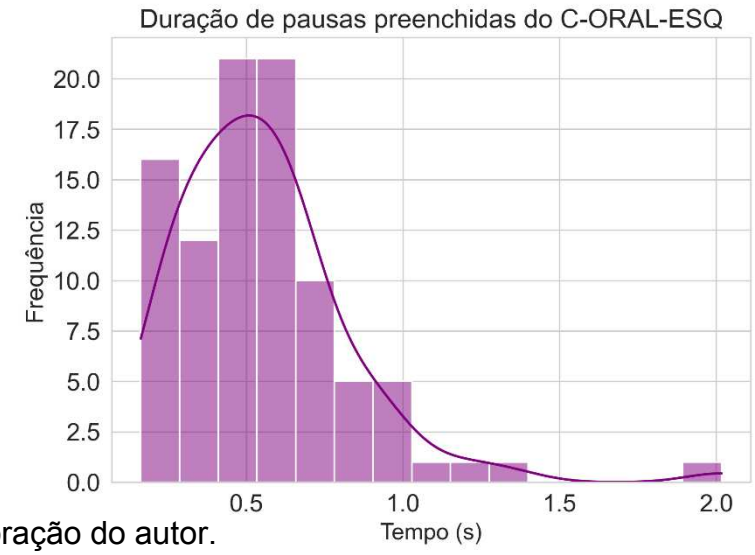
[38]Os áudios MED_003, MED_002_1 e MED_011 apresentaram problemas técnicos no arquivo gerado alinhamento original e não foi possível medir suas pausas preenchidas na ocasião. Dessa forma, a análise desse fenômeno desconsidera esses três áudios.

Fonte: Elaboração do autor

Em média, foram realizadas 7,30 pausas preenchidas por áudio (std = 4,84). Os pacientes com mais hesitações foram os de MED_029_2 e MED_007, e os menos disfluentes nesse sentido foram MED_008, MED_009. Chama a atenção a baixa quantidade de pausas preenchidas desses dois últimos.

O histograma da Figura 22 permite visualizar a duração das pausas preenchidas encontradas no C-ORAL-ESQ. Em relação a duração dessas pausas, é possível observar que as pausas preenchidas dos pacientes se concentra entre 500 e 750 s, com uma longa cauda à direita (média = 565 ms, std= 0,296 e mediana = 528 ms), com distribuição assimétrica negativa. Esse resultado é bastante similar ao encontrado na comparação entre pacientes e não pacientes e discutido na subseção 5.2.1.

Figura 22 – Duração de pausas preenchidas do C-ORAL-ESQ.

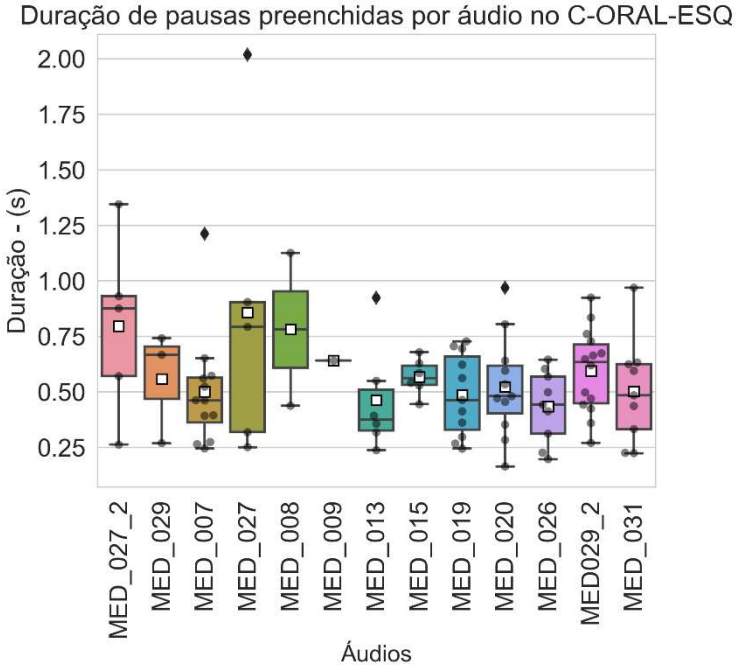


Fonte: Elaboração do autor.

Na Figura 23, é possível ver a duração das pausas preenchidas por paciente. Da esquerda para a direita, do áudio MED_013 adiante, observa-se que, de fato, a média da duração, dada pelo quadradinho branco nas caixinhas, se mantém entre 500 e 550 ms. As exceções ficam por conta de quatro áudios, a saber, MED_027, MED_027_2, MED_008 e MED_009. Desses, os dois primeiros são do mesmo paciente, mas em consultas diferentes O enorme desvio padrão de MED_027_2, dado pelo tamanho do parafuso que sai das caixinha, permite sugerir que a média desse paciente é artificialmente inflada, e a pouca quantidade de pontinhos pretos, que marcam a ocorrência de cada pausa, sugerem que se trata de um resultado pouco generalizável o fato de sua média estar acima dos outros pacientes. No caso de

MED_008 e MED_009, o principal fator que infla a duração dessas pausas é sua baixa representatividade, pois o primeiro tem 2 e o segundo apenas 1 pausa preenchida.

Figura 23 – Duração de pausas preenchidas por áudio

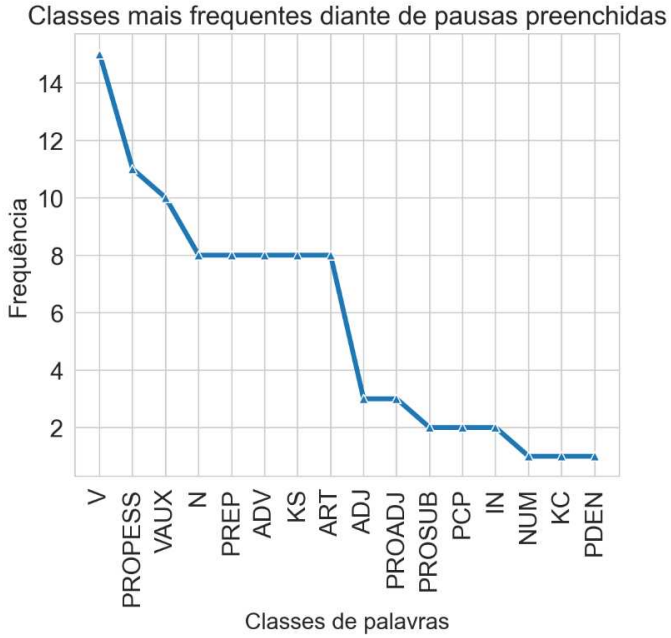


Fonte: Elaborado pelo autor.

4.6.1 Pausas preenchidas e classes de palavras

A maioria das pausas preenchidas do C-ORAL-ESQ ocorreu diante de verbos, seguidos por pronomes pessoais, verbos auxiliares e substantivos, como é possível visualizar na Figura 24.

Figura 24 – Frequência de pausas preenchidas diante de classes gramaticais.

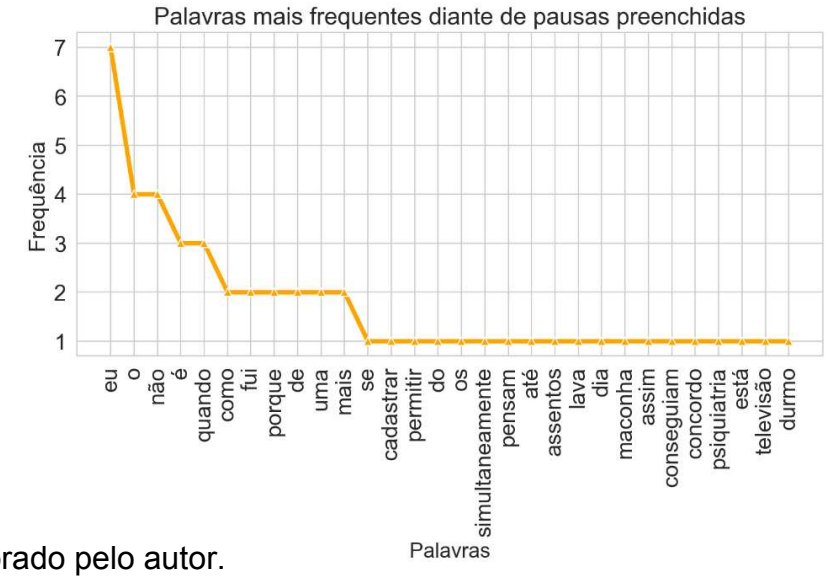


Fonte: Elaborado pelo autor.

Dessa forma, classes lexicais, como verbos, substantivos e advérbios, dividem os postos de maiores frequências com classes gramaticais, como pronomes pessoais, verbos auxiliares e preposições

Na Figura 25, com exceção de um advérbio (não), é possível observar que as palavras mais frequentes diante de pausas foram as pertencentes a classes gramaticais (eu, o, quando) e há uma longa linha reta de verbos que ocorrem apenas uma vez. Chama a atenção a frequência de pausas preenchidas diante do pronome pessoal *eu*.

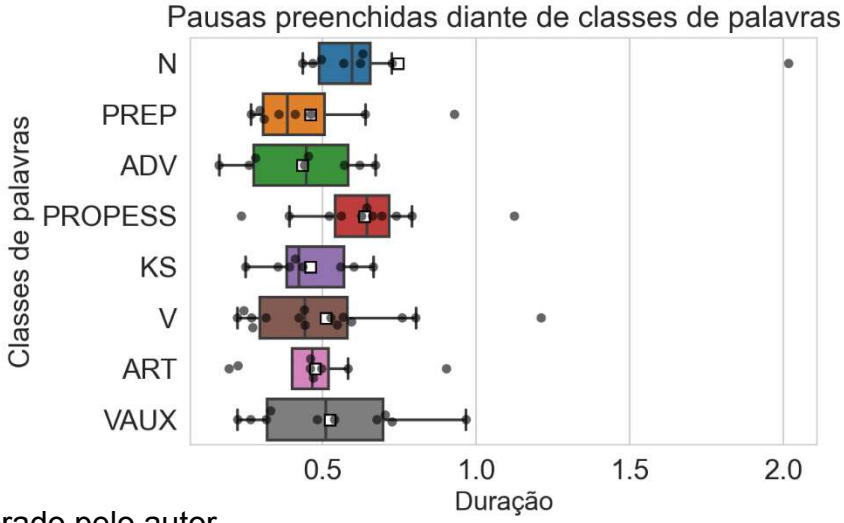
Figura 25 – Frequência de palavras diante de pausas preenchidas no C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

Se consideradas as classes de palavras, é possível observar diferenças em relação à duração de pausas preenchidas, tal como pode ser visto na Figura 26.

Figura 26 – Duração de pausas preenchidas diante de classes de palavras no C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

. Observa-se que as maiores médias de duração de pausas preenchidas ocorre diante de substantivos – este com um grande outlier à direita, que puxa a média para cima - e pronomes pessoais. A mediana , que não é afetada por esse dado destoante, de fato permite visualizar que as pausas desses pacientes foram maiores diante dessas duas classes (597 e 645 ms, respectivamente). Entretanto, não há diferença estatística relevante entre a duração das pausas diante dessas classes (p = 0,261 no teste H de Kruskal Wallis).

Em futuros trabalhos, é possível avançar essa descrição para o nível do sintagma e verificar se essas pausas ocorrem em suas fronteiras, por exemplo.

4.7 Panorama do léxico do C-ORAL-ESQ

Esta seção descreve sucintamente o léxico e as classes gramaticais mais frequentes no C-ORAL-ESQ, bem como alguns detalhes específicos dessas últimas.

4.7.1 Colocações linguísticas

As colocações foram medidas de duas formas. Na primeira, bem sucinta, as colocações foram dadas após o tratamento dos dados e aplicação do *collocations*, da NLTK. A segunda forma, mais completa, permite visualizar 14.082 combinações para bigramas e 62.975 combinações para trigramas, ambas ranqueadas em ordem decrescente por probabilidade de ocorrência no C-ORAL-ESQ. Esta última contagem requisitou os procedimentos elencados na documentação da NLTK[39] e foge do escopo desta tese explicá-los detidamente. Em linhas gerais, todas as palavras do C-ORAL-ESQ são dispostas uma ao lado da outra, duas de cada vez quando bigramas e três quando trigramas. Dessa forma, é possível contar a frequência de ocorrência desses pares ou trios e calcular sua probabilidade de ocorrência. Os resultados do primeiro procedimento podem ser vistos no Quadro 4.

Algumas repetições de palavras são bigramas esperados na fala, tais como *hum hum,* bastante frequentes em Comentários curtos; ou *tipo assim*, bastante frequentes no C-ORAL-ESQ e podem funcionar como Parentéticos; e construções

39

como *falou assim* , que canonicamente insere um discurso reportado e podem atuar como Introdutores Locutivos.

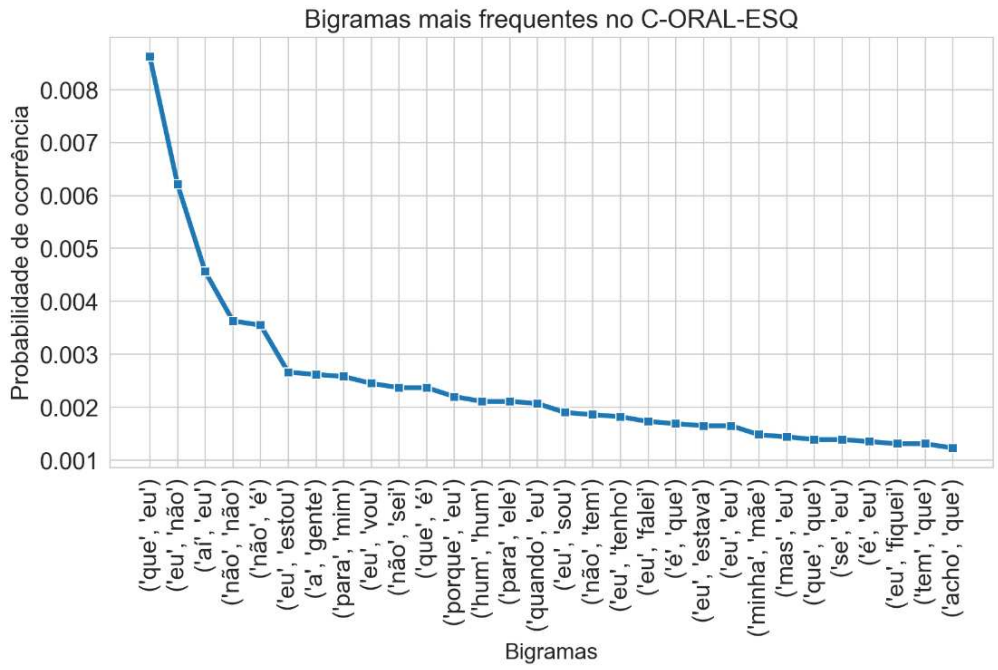
Quadro 4 – Principais colocações linguísticas no C-ORAL-ESQ

Colocações linguísticas
hum hum; para mim; minha mãe; não sei; meu pai; para ele; todo mundo; minha irmã; acho que; por causa; investe investe; ela falou; belo horizonte; por exemplo; tipo assim; falou assim; minhas filhas; sim senhora; doutora luisa; não tem

Fonte: Elaboração do autor

Já a extensa lista de bigramas e trigramas[40] permite ver maiores nuances de acordo com a probabilidade de ocorrência. Os bigramas mais frequentes podem ser visualizados na Figura 27, enquanto os trigramas na Figura 28. É importante destacar que, quanto mais frequentes são essas combinações, mais intuitivas linguisticamente elas são. O fato de combinar todas as palavras do corpus em combinações de 2 ou 3, entretanto, nem sempre gera ocorrências reconhecíveis por um falante nativo. Por isso, as combinações mais que gerem maior estranheza são um tipo de termômetro linguístico para sua baixa frequência.

Figura 27- Bigramas mais frequentes no C-ORAL-ESQ.

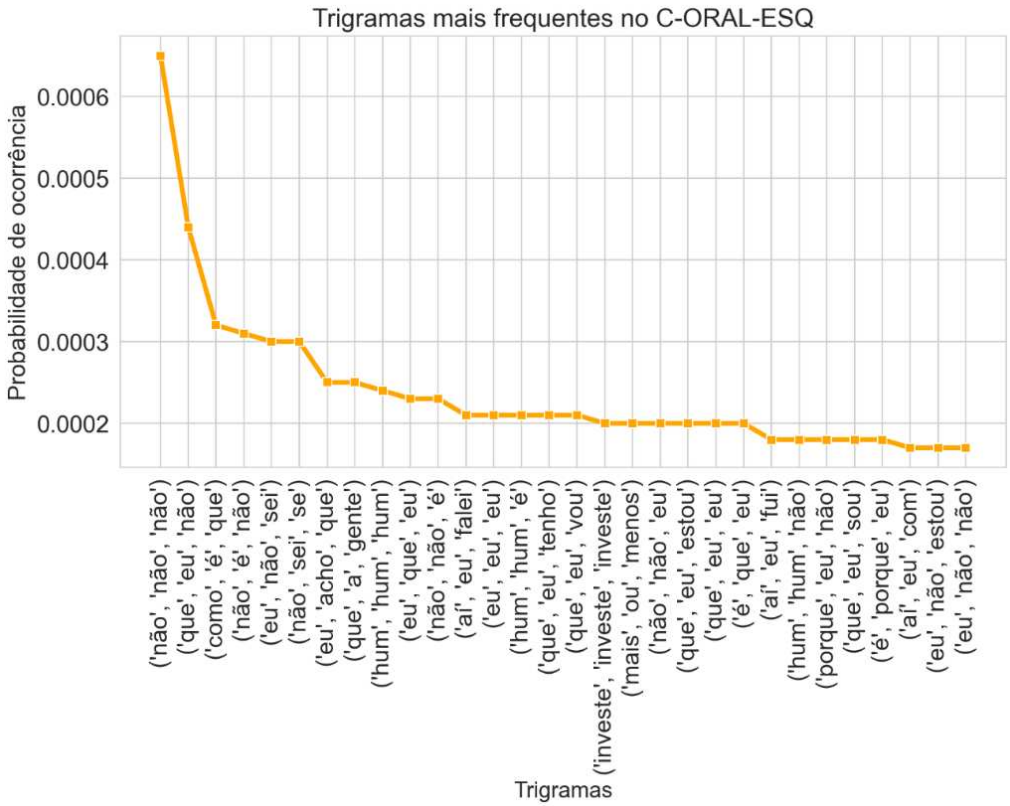


Fonte: Elaboração do autor.

[40]https://drive.google.com/drive/folders/1vLEwgA_CxJhhuna_awHrXhhUZMbqYfDQ?usp=sharing

Há muitas colocações que se relacionam de alguma forma à negação e à palavra “não” tanto na de bigramas (“eu não”, “não é”, “não sei”) quanto na de trigramas (“aí eu não”, “não é não”). Também é possível observar que muitas dessas colocações tem o pronome pessoal “eu”, palavra mais frequente do corpus com stopwords, quer como em possíveis orações subordinadas (ex: ele disse *que eu não* estava bem; acho *que eu vou,* com trigramas em destaque), quer como sujeito de orações (*aí eu falei*, *eu acho que*).

Figura 28 – Trigramas mais frequentes no C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

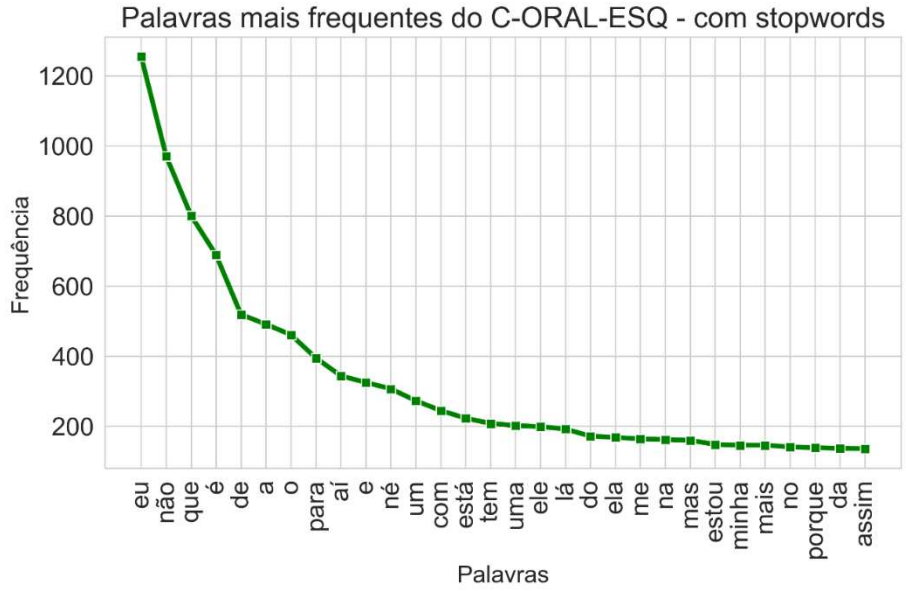
4.7.2 Palavras mais frequentes

Apesar de uma primeira filtragem para retirar stopwords previstas pela biblioteca em português do Spacy, a maioria das palavras mais frequentes no corpus são claramente de classes gramaticais, conforme é possível ver na Figura 29.

Das trinta primeiras palavras mais frequentes, as únicas lexicais é o advérbio *não,* sua forma contraída *né* (não + é) e o também advérbio *lá*. Todas as outras

palavras são gramaticais. O predomínio dessas palavras, chamadas de stopwords, é esperado em uma transcrição. Para observar palavras de maior conteúdo semântico, como verbos lexicais, substantivos e adjetivos, é necessário retirá-las, ou pelo menos a limpar a maioria delas.

Figura 29 – Palavras mais frequentes do C-ORAL-ESQ com as stopwords.



Fonte: Elaborado pelo autor.

A Figura 30 traz uma nuvem de palavras[41], em sua maioria lexicais, com as palavras mais frequentes do C-ORAL-ESQ. O objetivo dessa visualização é ter uma visão esquemática principalmente das palavras de conteúdo, como substantivos, verbos e adjetivos, discutidas nas seções a seguir. Na Figura 30, moldada pelo contorno dos protagonistas de uma das pinturas de *A cat and her kittens*, na qual uma mamãe gata segura seu bebê gatinho, de Louis Wain[42], as palavras mais frequentes aparecem em fonte maior.

Como foi realizada uma primeira filtragem de stopwords a partir da lista fornecida pela biblioteca Spacy, já é possível observar mais palavras de conteúdo. Em linhas gerais, as palavras mais recorrentes são as utilizadas na rotina dos pacientes, tanto no âmbito familiar (*casa, dia, mãe, pai, irmã, hora)* quanto no consultório (*doutora, doutor, remédio*). Substantivos que ocasionalmente figuraram como assunto também aparecem, como Olanzapina, cigarro e maconha, mas substantivos mais genéricos, como os já mencionados, parecem despontar como mais frequentes do

[41]Ver instruções para usar o Programa 2, que descreve o léxico e pode gerar nuvens de palavras moldáveis como essa.
[42]Pintor com esquizofrenia e mundialmente conhecido por pintar inúmeros gatos, muitas vezes em situações tipicamente humanas: https://www.wikiart.org/pt/louis-wain

que os mais específicos. Observa-se baixa densidade de adjetivos, como *velha,* e *esquisita*, além de grande ocorrência de formas verbais do verbo *falar*.

Figura 30 – Nuvem de palavras do C-ORAL-ESQ na forma de *A cat with her kittens*, de Louis Wain.

Foto: Elaborado pelo autor.

4.7.3 Classes de palavras mais frequentes

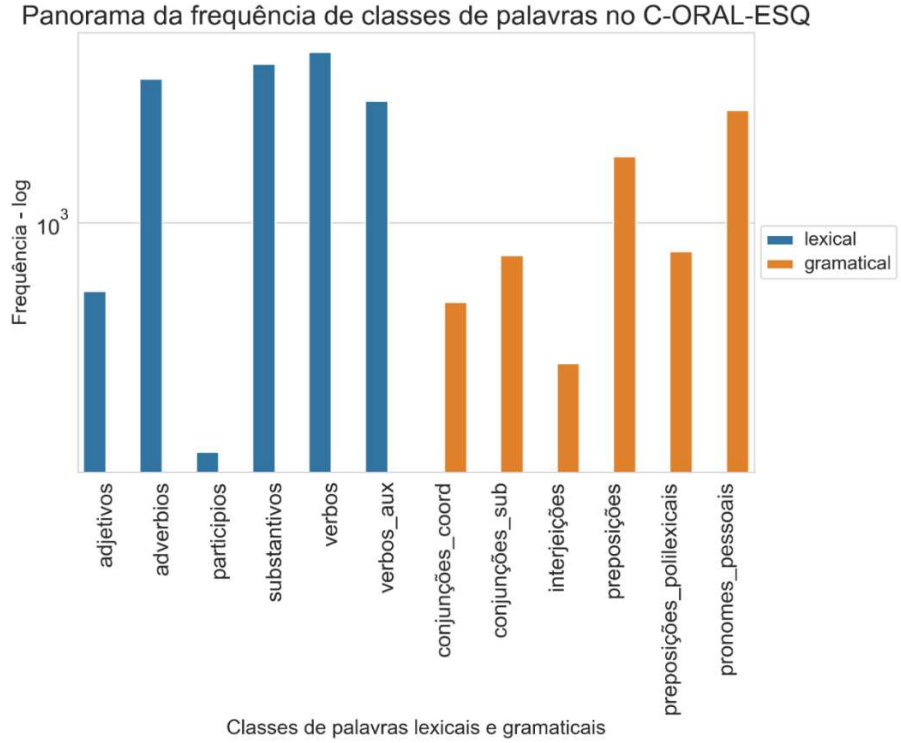
Optou-se por representar apenas algumas das classes de palavras – todas lexicais – e disponibilizar as outras para consulta e visualização[43].

Na Figura 31, é possível verificar que advérbios, substantivos e verbos são as classes lexicais mais frequentes do corpus. Nas classes gramaticais selecionadas,

[43]A grande maioria das visualizações deste capítulo são geradas pelo Programa 2, conforme instruções no Anexo B.

preposições simples, pronomes pessoais e verbos auxiliares foram as mais frequentes.

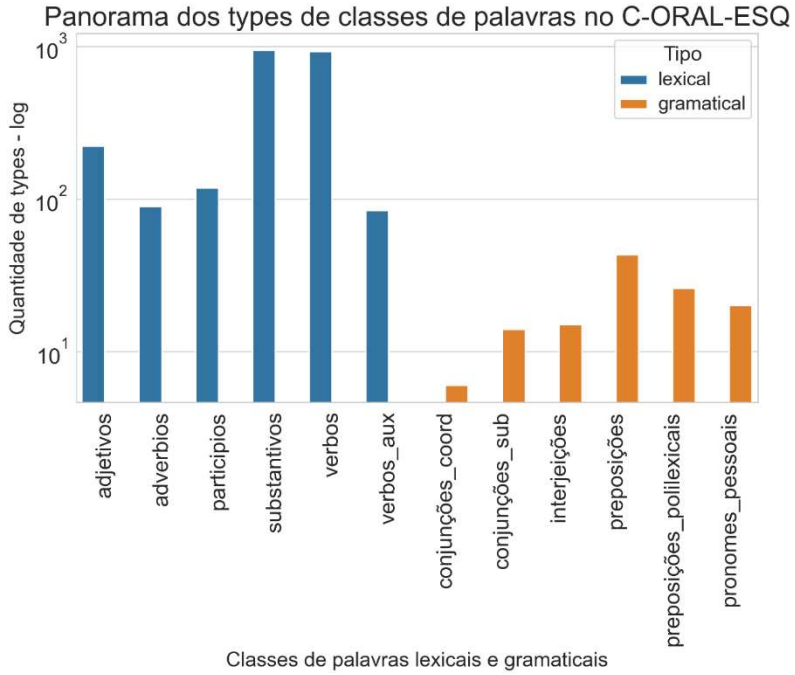
Figura 31 – Classes de palavras mais frequentes no C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

A diversidade dos types é naturalmente maior nas classes variáveis e lexicais. Isso pode ser visto na Figura 32, na qual é possível observar que substantivos, verbos e adjetivos lideram esse ranking.

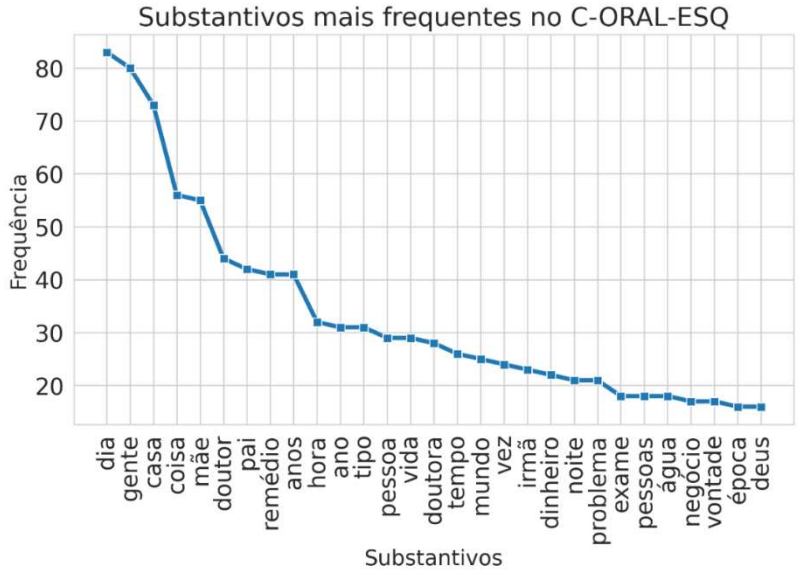
Figura 32 – Types de classes de palavras no C-ORAL-ESQ.



4.7.3.1 Substantivos

Conforme já mencionado, a maioria dos substantivos utilizados pelos pacientes dizem respeito à rotina (dia, hora, tempo), tanto de casa ou família (casa, mãe, pai) quanto de consultório (doutor, doutora, remédio, problema, exame). Isso pode ser visto na Figura 33.

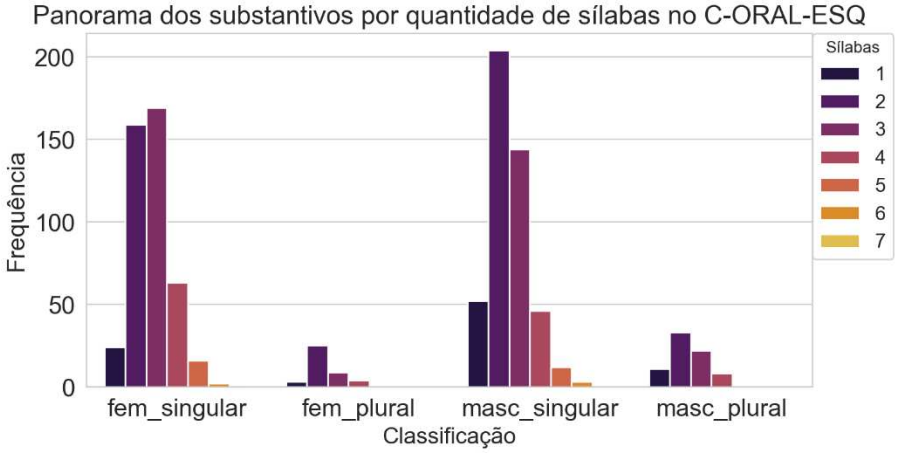
Figura 33 – Substantivos mais frequentes no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Entre as mais de 900 ocorrências, predominam os substantivos masculinos no singular, com duas a três sílabas, seguidos pelos femininos no singular, com três a duas sílabas, respectivamente (Figura 34).

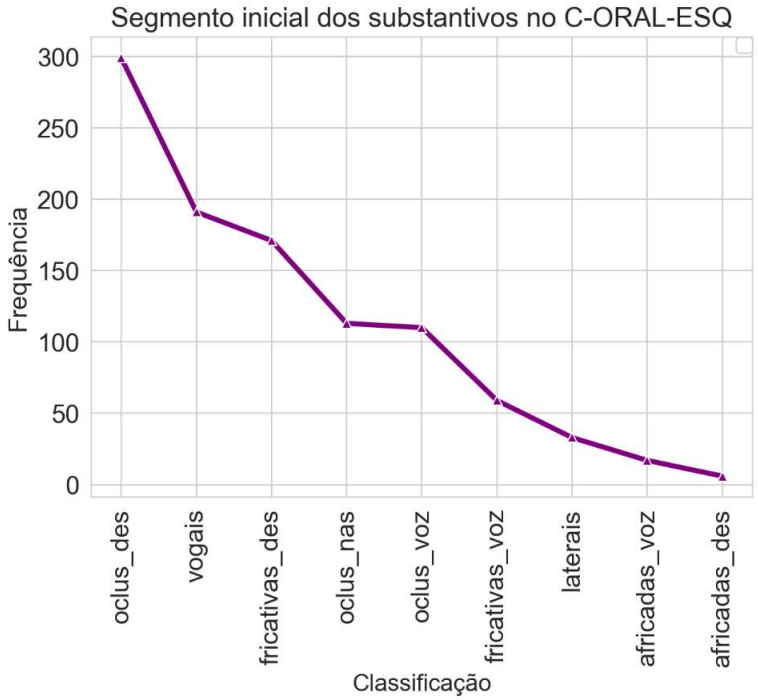
Figura 34 – Classificação morfológica e silábica dos substantivos do C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Esses substantivos, em sua maioria, começam com uma oclusiva desvozeada ou uma vogal (Figura 35), ao passo que os menos frequentes começam com africadas, independemente do vozeamento.

Figura 35 – Classificação fonética do segmento inicial dos substantivos do C-ORAL-ESQ.

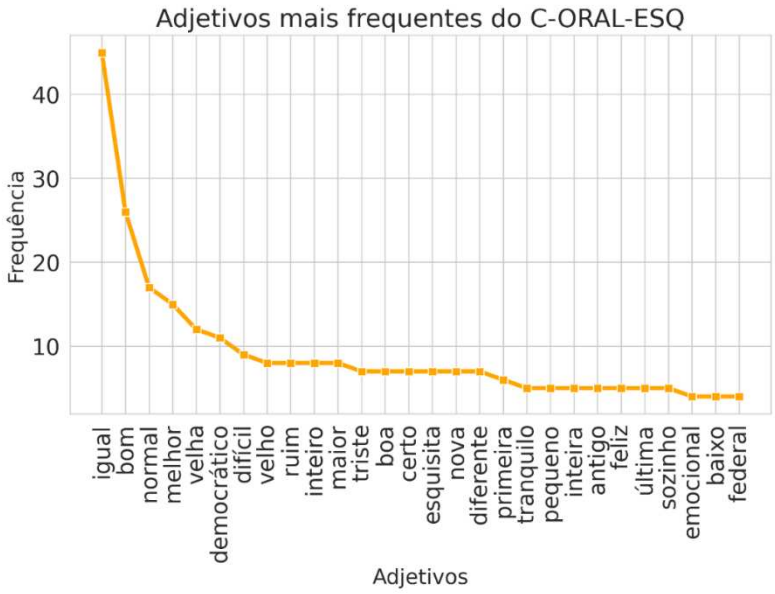


Fonte: Elaborado pelo autor.

4.7.3.2 Adjetivos

A figura a seguir ilustra os adjetivos mais frequentes do C-ORAL-ESQ.

Figura 36 – Adjetivos mais frequentes do C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Tradicionais modificadores de substantivos, os adjetivos mais frequentes são mais positivos do que negativos, ao passo que os abaixo de dez ou menos são mais negativos (difícil, ruim, sozinho), conforme é possível ver na Figura 36.

Observar esse padrão poderia ajudar na identificação automática pelo léxico sobre como pode estar o humor desses pacientes dentro do consultório, seja quando falam de sua rotina, como claramente acontece no começo das interações, seja quando falam com mais desenvoltura de si, em stanzas mais longas, com o desenrolar da consulta. Nas Figuras 37 e 38 a seguir, é possível visualizar detalhes morfológicos e fonéticos desses adjetivos.

Figura 37- Detalhes morfológicos e silábicos do C-ORAL-ESQ

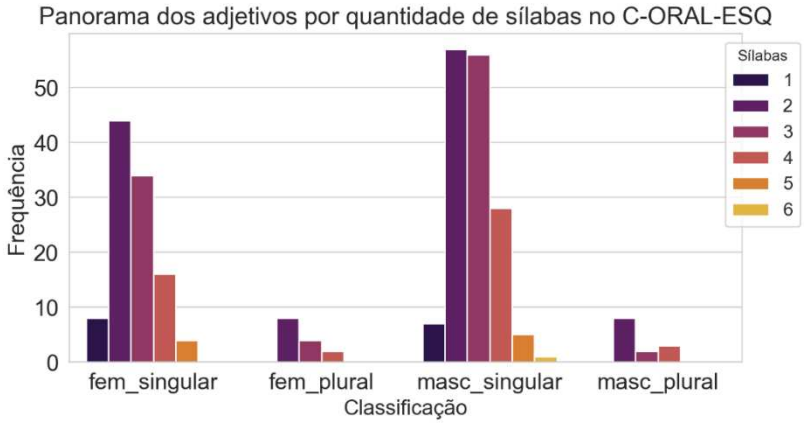
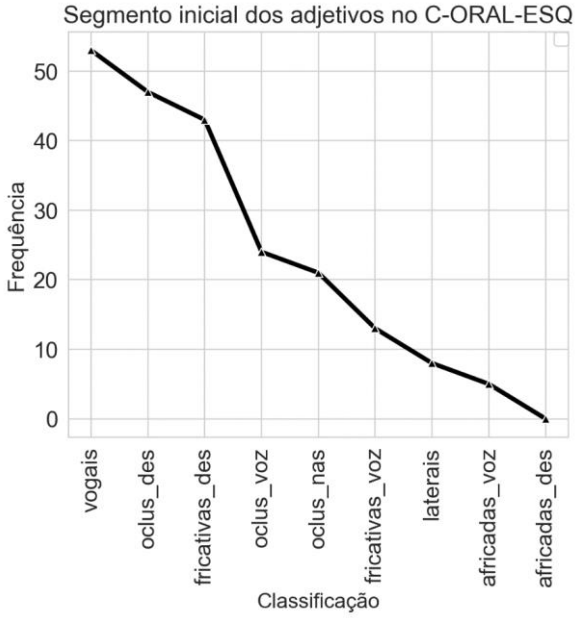


Figura 38 – Classificação fonética do segmento inicial dos adjetivos do C-ORAL-ESQ.



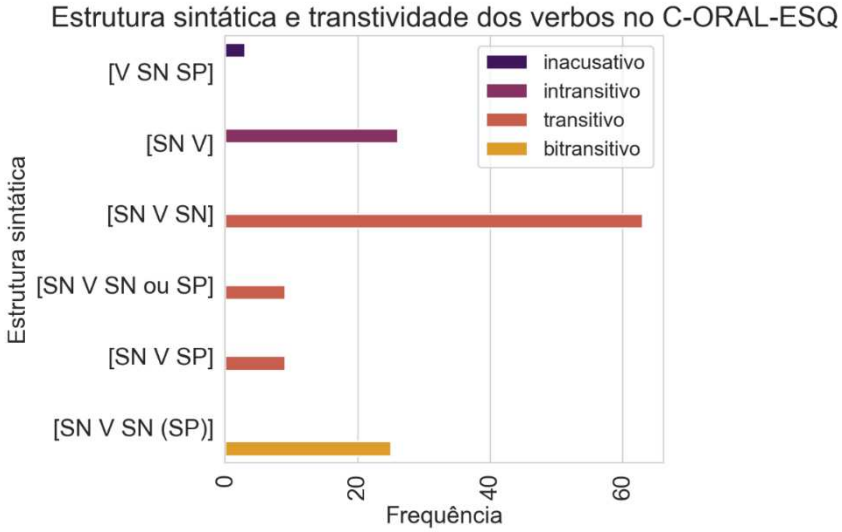
Fonte: Elaborado pelo autor.

É possível observar que a grande maioria desses adjetivos era do gênero masculino singular, com 2 a três sílabas, seguidos pelos femininos no singular, também com duas a três sílabas, e começam por vogais ou oclusivas desvozeadas, respectivamente.

4.7.3.3 Verbos

Como é possível observar na Figura 39, a maioria dos verbos do C-ORAL-ESQ são transitivos diretos [SN V SN], tais como *entender, ligar*, *passar*, seguidos pelos inacusativos [SN V] (*falar, trabalhar, conversar*) e bitransitivos [SN V SN (SP)] (*parar, colocar, esconder*).

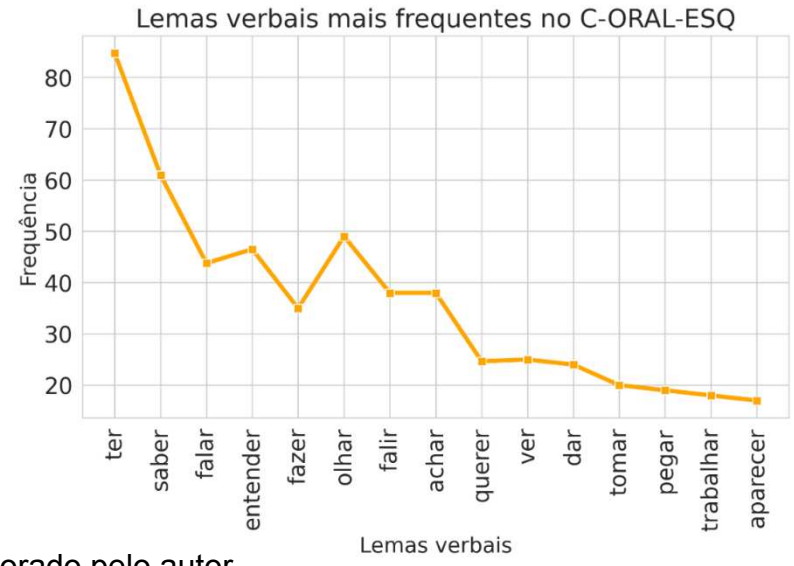
Figura 39 – Estrutura sintática dos verbos no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Os lemas verbais mais frequentes são de verbos existenciais (*ter*), verbos de mudança de estado mental (*saber, entender*) e o verbo de atividade *falar*. Em sua maioria, a Figura 40 ilustra verbos mais lexicais (*saber, falar, entender*) do que verbos mais gramaticais (*ser, estar, ficar, deixar*). Na atual versão do etiquetador construído para viabilizar as discussões propostas nesta tese, ainda há uma oscilação entre a classificação do verbo *ter* como verbo lexical, como em *ele tinha um carro* e como verbo auxiliar, como em *ele tinha comprado um carro*. A etiquetagem de verbos auxiliares, que já é realizada, é algo criado para esta tese e não consta nas etiquetas do Mac-Morpho. Implementações já em curso permitem resolver essa questão para não inflar artificialmente a contagem dos lemas mais frequentes de verbos que têm conteúdo semântico mais autônomo do que verbos auxiliares.

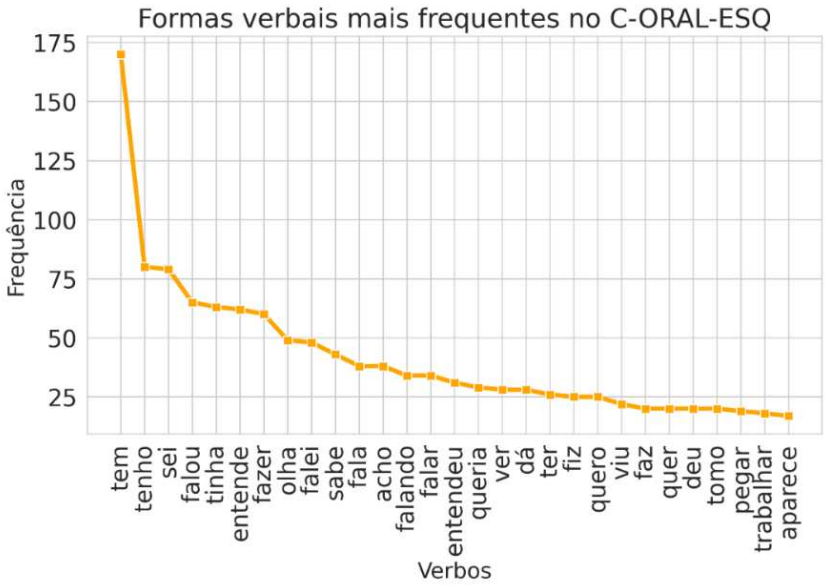
Figura 40 – Lemas dos verbos mais frequentes no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

As formas verbais mais frequentes dos pacientes, na Figura 41, acompanham essa distribuição dos lemas e são complementadas pela distribuição do tempo e modo verbal da Figura 42.

Figura 41- Formas verbais mais frequentes no C-ORAL-ESQ.

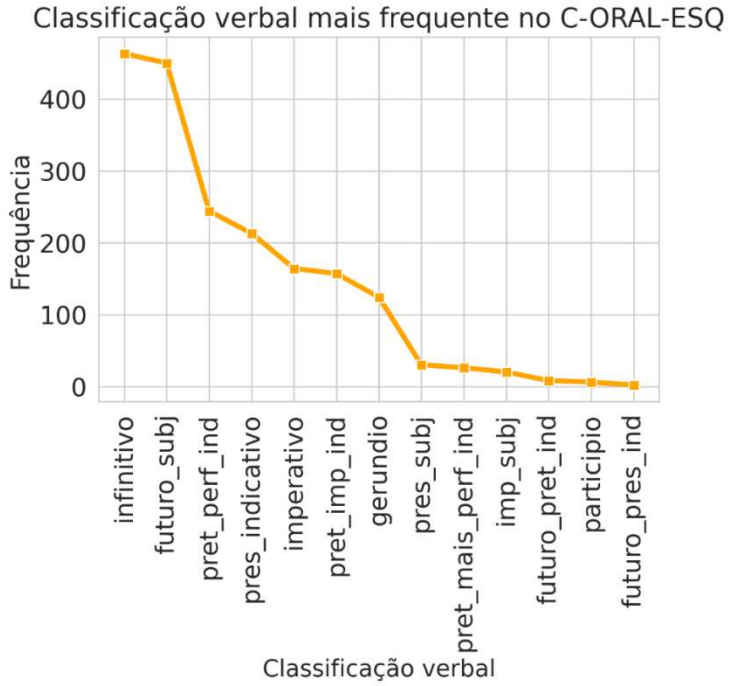


Fonte: Elaborado pelo autor.

Na Figura 42, observa-se o predomínio do infinitivo (*trabalhar, passar, entrar*), o qual dispõe de muitos types – neste caso, verbos – mais à direita, que ocorrem pouco ou apenas uma vez. Também se destacam o futuro do subjuntivo – comumente

utilizado com o verbo *ir* (*se eu for*) e, em muitos casos, contabilizado da mesma forma que um infinitivo devido à análise lexical por palavra do DELAF[44].

Figura 42 – Tempo e modo verbal mais frequentes no C-ORAL-ESQ



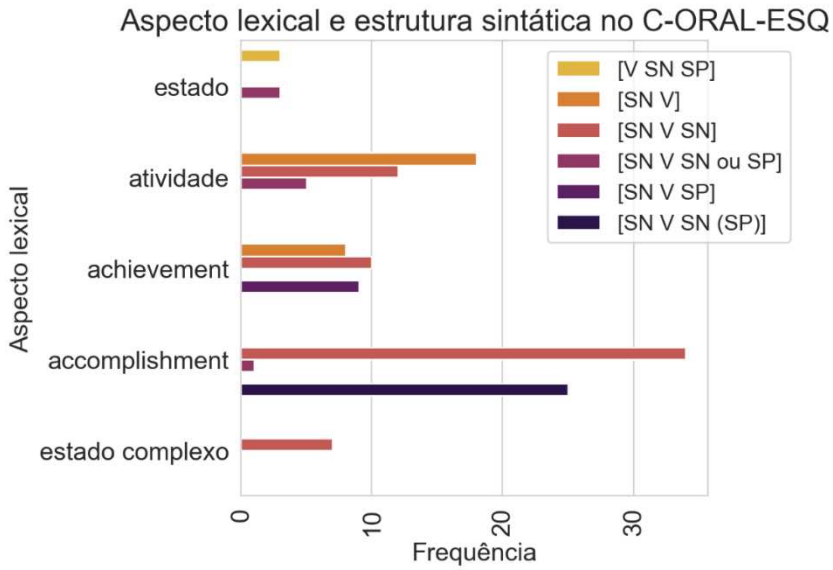
Fonte: Elaborado pelo autor.

Além disso, foram muito frequentes formas do Pretérito Perfeito do Indicativo, principal responsável por atualizar o aspecto gramatical do verbo e, em termos discursivos, mover a narrativa contada pelos pacientes durante a consulta para um domínio cognitivo novo. De fato, a ocorrência do Pretérito Perfeito do Indicativo – que é seguido de perto em termos de frequência pelo Imperfeito do Indicativo – pode sugerir a presença de sequências de tipologia mais narrativa e a presença de situações que os pacientes usualmente contam em stanzas mais longas.

A maioria desses verbos têm o aspecto lexical (cf. CANÇADO et AMARAL, 2016) de accomplishment (*parar, ligar, colocar*), são transitivos (ex: *liguei* a TV, *machuque*i o joelho) e bitransitivos (ex: *colocou* o livro na prateleira; *enfiou* a agulha no braço). A esses seguem verbos de atividade (*passar, mexer, escovar, andar*) que são intransitivos (ex: ele *falou*; ele *trabalhou*) e transitivos (ex: o médico *limpou* o consultório; ela *escovou* o cabelo). Essas e outras ocorrências podem ser visualizadas na Figura 43.

[44]A maioria dos verbos classificados no infinitivo é classificada no futuro do subjuntivo no DELAF.

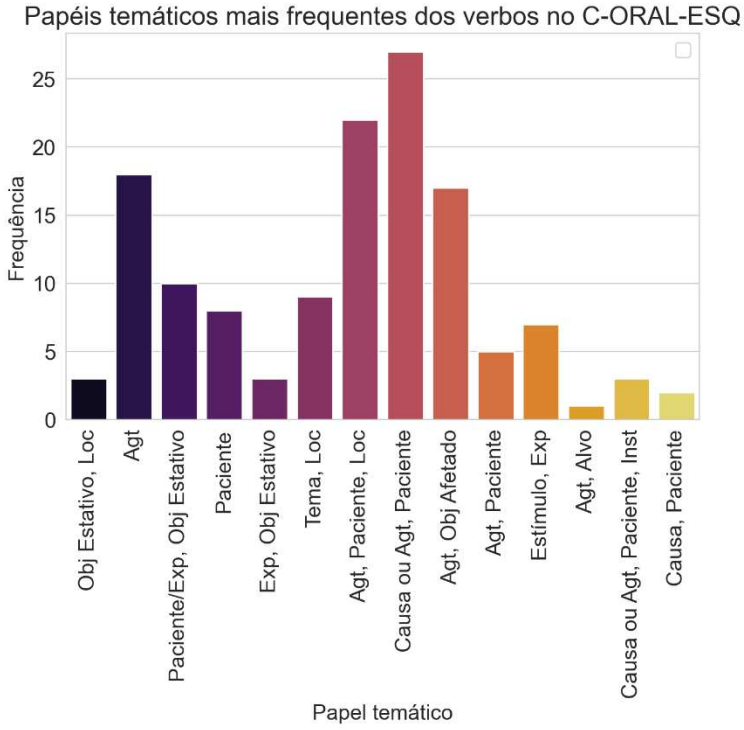
Figura 43- Aspecto lexical e estrutura sintática dos verbos do C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Já os papéis temáticos mais requisitados por esses verbos no C-ORAL-ESQ podem ser vistos na Figura 44.

Figura 44 – Papéis temáticos mais frequentes da grade temática dos verbos do C-ORAL-ESQ.



Fonte: Elaborado pelo autor.

Na Figura 44, é possível obserbvar que os papéis temáticos mais frequentes são os tradicionalmente envolvidos em verbos de Causação, isto é, Causa ou Agente

e Paciente (ex: ele *quebrou* o vaso) e os canonicamente exigidos por verbos bitransitivos, ou seja, Agente, Paciente e Locativo (ex : o médico *colocou* a receita na mesa).

Do ponto de vista fonológico, a maioria desses verbos têm entre 2 e 3 sílabas (Figura 45), ao passo que seus segmentos iniciais mais frequentes foram oclusivas desvozeadas (formas verbais de *ter* e *querer*, principalmente) e vogais (formas verbais de *achar*, *acontecer* e *ajudar*, entre outros), conforme ilustrado pela Figura 46.

Figura 45- Quantidade de sílabas dos verbos do C-ORAL-ESQ.

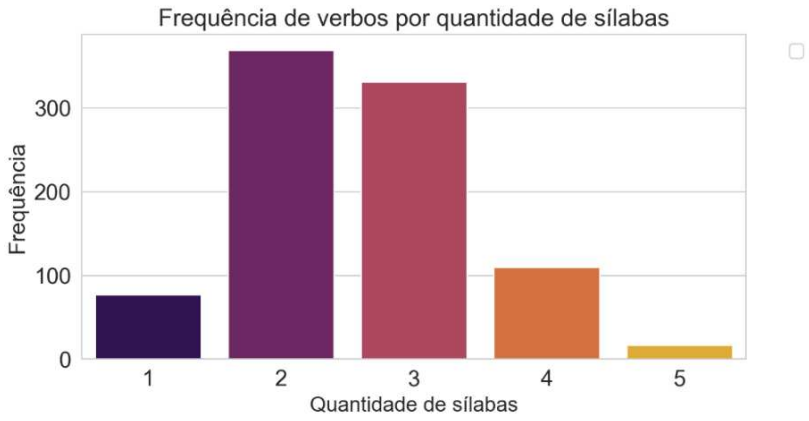
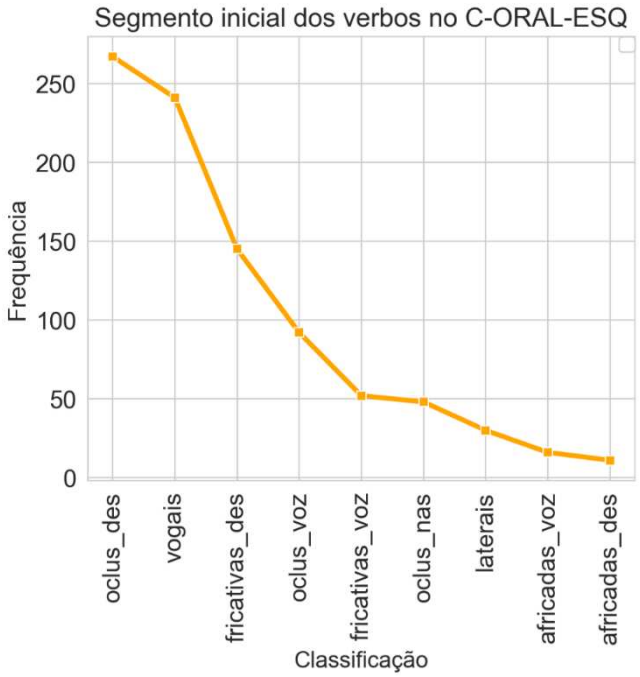


Figura 46 – Classificação fonética do primeiro segmento dos verbos do C-ORAL-ESQ



Fonte: Elaborado pelo autor

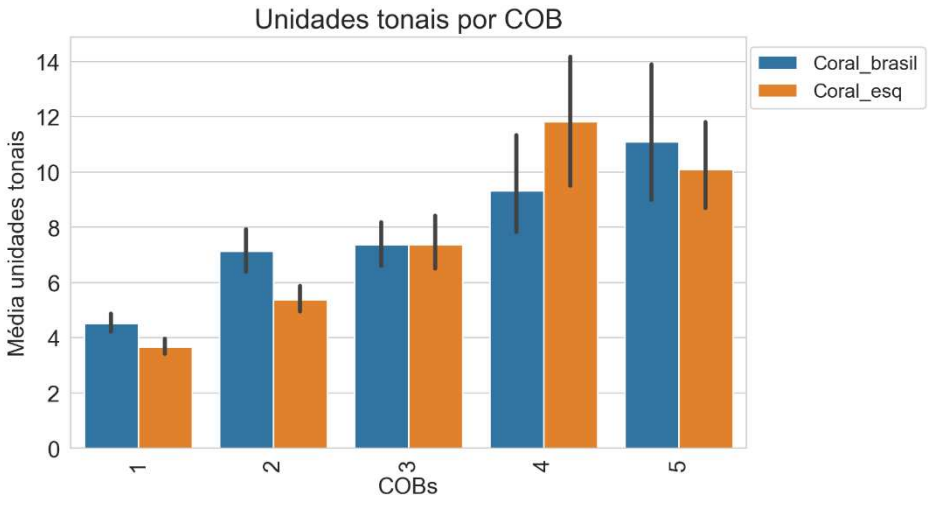
5 COMPARAÇÃO DE PADRÕES INFORMACIONAIS E MEDIDAS DA FALA ENTRE C-ORAL-ESQ E C-ORAL-BRASIL

Este capítulo apresenta e discute os resultados encontrados a partir da comparação entre pacientes com esquizofrenia do C-ORAL-ESQ e participantes do C-ORAL-BRASIL. Questões estruturais dos enunciados e medidas da fala são discutidas na seção 5.1. A seção 5.2 compara as disfluências entre os dois grupos, tais como pausas e enunciados interrompidos e, a seguir, é feito um breve sumário dessas duas primeiras seções em 5.3. A comparação entre os padrões informacionais entre pacientes e não pacientes é realizada na seção 5.4, a qual também é seguida por um sumário para auxiliar o leitor na seção 5.5. Nas seções 5.6 são discutidos dois tipos de Escansão, com e sem retractings, e, por fim, as unidades dialógicas são trazidas à tona na seção 5.7.

5.1 Unidades tonais, quantidade de palavras e duração

Pacientes com esquizofrenia realizaram menor quantidade de unidades tonais em stanzas mais curtas, de 1 a 2 COBs, com relevância estatística (p = 0.0001 nos dois casos) e maior quantidade nas de 3 e 5 COBs. Isso pode ser visualizado na Figura 47.

Figura 47 – Comparação de unidades tonais por COB.



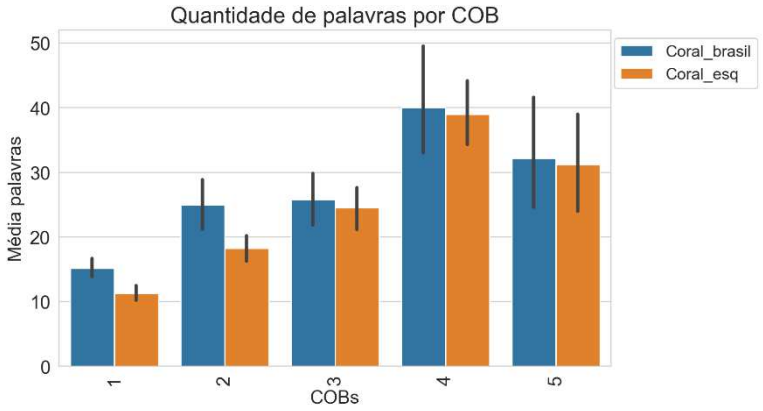
Fonte: Elaborado pelo autor.

Como existe o pareamento entre unidade tonal e unidade informacional, exceto em casos de Escansão e Comentários Múltiplos (cf. 2.2.1), é de se esperar que o

grupo que apresentasse menor quantidade de unidades textuais também tivesse menor quantidade de unidades tonais. Nas amostras mais relevantes, portanto, as stanzas dos pacientes foram mais curtas nesse sentido.

As stanzas dos pacientes possuem menos palavras que as dos não pacientes, com relevância estatística nas amostras de 1 e 2 COBs (p = 0.0008 e p =0.006, respectivamente). Em uma mesma quantidade de COBs, portanto, pacientes realizam seu padrão informacional com menor quantidade de unidades tonais e menor quantidade de palavras. Isso pode ser visto na figura 48.

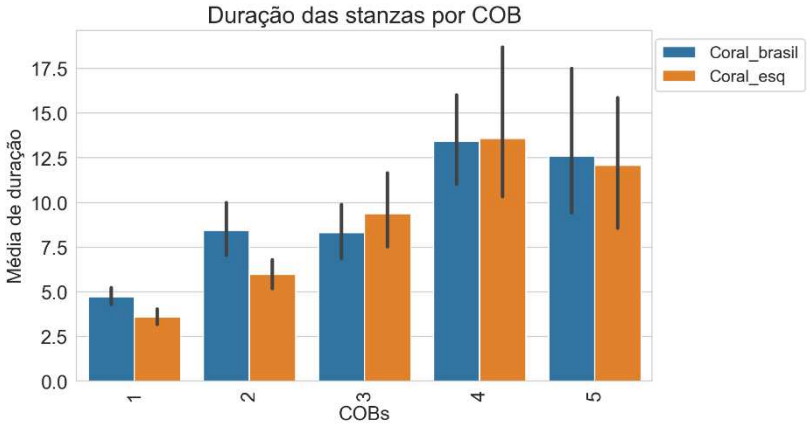
Figura 48 - Comparação de quantidade de palavras por COB.



Fonte: Elaborado pelo autor.

A duração das stanzas dos pacientes é mais curta na maioria das amostras, exceto na de 3 COBs, conforme é possível ver na Figura 49.

Figura 49 - - Comparação da duração das stanzas por COB.

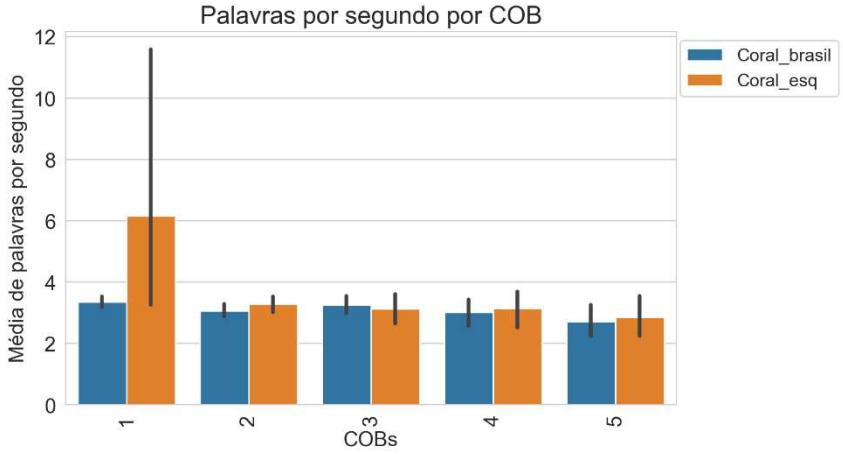


Fonte: Elaborado pelo autor.

A comparação entre quantidade de palavras por segundo (Figura 50) tem o objetivo apenas de ilustrar, em alguma medida, certo tipo de fluência dos participantes. Os resultados mostram que essa variável tem frequência relativamente

semelhante nos dois grupos, exceto na amostra de 1 COB, na qual os pacientes realizaram suas stanzas mais rapidamente, mas sem relevância estatística na comparação com o grupo do C-ORAL-ESQ.

Figura 50 - Comparação da quantidade de palavras por segundo por COB.



Fonte: Elaborado pelo autor.

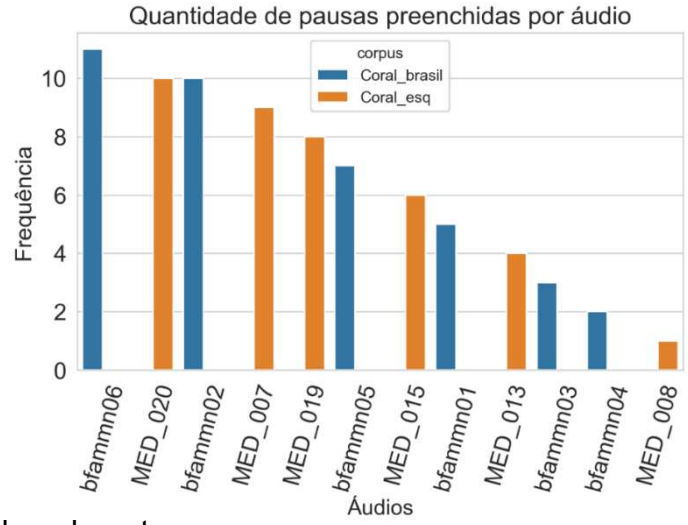
5.2 Disfluências

Nesta seção são detalhadados resultados obtidos na comparação de pausas preenchidas, pausas silenciosas, enunciados interrompidos e retractings entre o grupo do C-ORAL-ESQ e o grupo do C-ORAL-BRASIL.

5.2.1 Pausas preenchidas

As 76 pausas preenchidas, 38 de cada corpora, possuem a distribuição apresentada na Figura 51. Observa-se uma distribuição bastante irregular em termos de frequência de ocorrência de pausas preenchidas entre os áudios. Dessa forma, não é viável uma comparação que considere apenas essa frequência, pois isso demandaria mais áudios e corpora mais compatíveis em termos de contextos e duração dessas interações. Afinal, é natural que haja mais hesitações em áudios mais longos e isso não necessariamente pode ser categorizado como uma disfluência. Por isso, é realizada uma comparação apenas entre a duração dessas pausas. Dessa forma, é comparada a duração entre 38 pausas preenchidas do C-ORAL-BRASIL e 38 do C-ORAL-ESQ.

Figura 51 - Quantidade de pausas preenchidas por áudio.



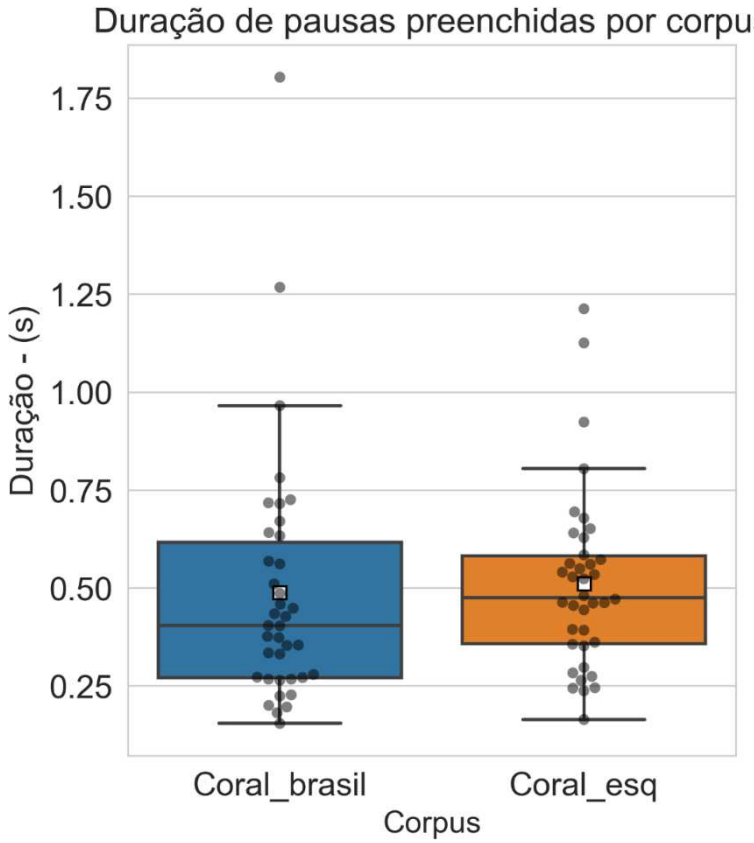
Fonte: Elaborado pelo autor.

Na Figura 52, é possível visualizar a distribuição de pausas preenchidas nos dois corpora. É possível observar que os pontinhos negros – que representam a duração de cada pausa dos corpora – se concentram mais acima em relação ao eixo y no C-ORAL-ESQ do que no C-ORAL-BRASIL. Por essa razão, a média observada no subcorpus de comparação é maior para pacientes (0,510 s, std = 0,227) do que não pacientes (0,487 s, std = 0,321).

A maior duração das pausas dos indivíduos com esquizofrenia também é atestada pela visualização da mediana, que corta transversalmente as caixinhas e é mais elevada em relação ao eixo y no C-ORAL-ESQ do que no C-ORAL-BRASIL. Ambos os corpora apresentaram dois outliers, os quais podem ser vistos pelos pontinhos pretos que estão acima de 1 segundo e fora do parafuso da caixa. Entretanto, essas diferenças observadas não são relevantes estatisticamente (p = 0.232 no teste U de Mann Whitney).

Conforme já adiantado em 4.6, a duração das pausas dessa amostra do C-ORAL-ESQ é bastante similar à utilizada no capítulo de descrição preliminar, apesar da amostra dessa última ter mais do que o dobro de pacientes. É necessário replicar essa pesquisa comparativa em larga escala, mas, a princípio, há uma aparente recorrência da duração média das pausas preenchidas dos pacientes, que é em torno de 500 ms. Entrentanto, como p > 0,05, não é possível refutar a Hipótese Nula 2 para essa variável, pois não há resultados estatisticamente relevantes que indiquem que pacientes com esquizofrenia realizem pausas maiores do que não pacientes.

Figura 52 - Comparação da duração das pausas preenchidas por corpus.



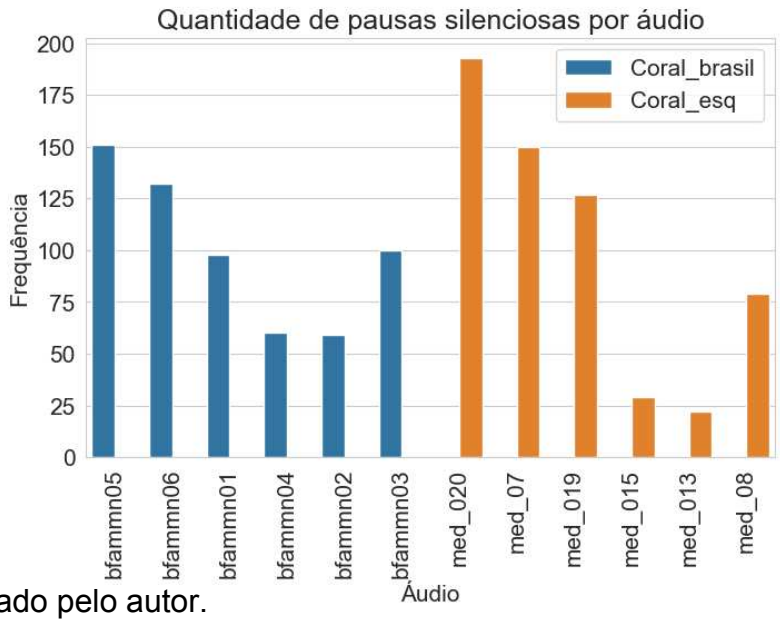
Fonte: Elaborado pelo autor.

Ademais, é necessário analisar individualmente a ocorrência de cada uma dessas pausas preenchidas – em futuros trabalhos de viés mais qualitativo - para verificar se essas pausas se tratam ou não, de fato, de disfluências. Não é incomum que pausas preenchidas sejam utilizadas para manter o turno de fala, uma vez que uma pausa silenciosa poderia sinalizar para outro falante – neste caso, o médico – de que o turno do paciente já poderia ser tomado.

5.2.2 Pausas silenciosas

Foram encontradas quantidades relativamente semelhantes de pausas silenciosas entre pacientes (600) e não pacientes (652). Apesar de certa homogeneidade entre grupos, a heterogeneidade entre a frequência de ocorrência desse fenômeno entre os áudios é grande, conforme pode ser visualizado na Figura 53.

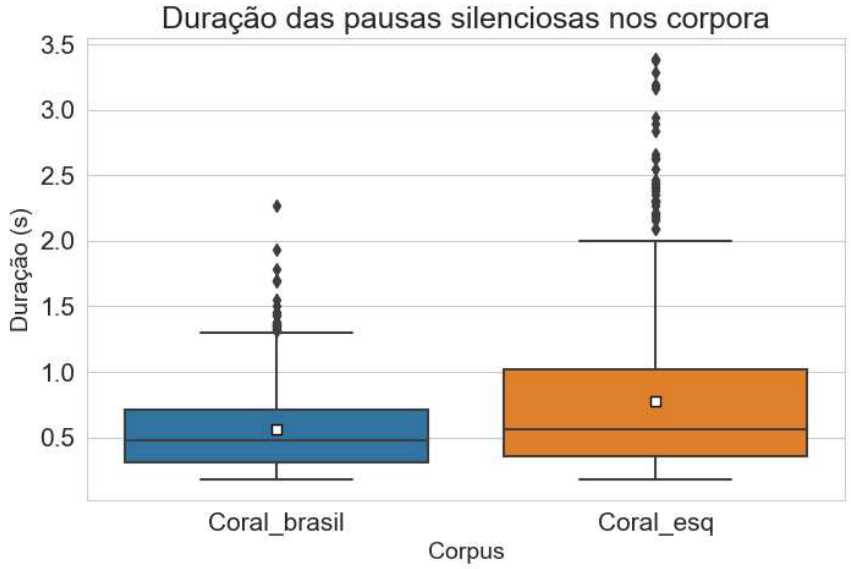
Figura 53 - Frequência de pausas silenciosas por áudio.



Elaborado pelo autor.

Após o balanceamento (cf. 3.11.6.2), a distribuição desse fenômeno em termos de duração nos dois corpora (n = 600 para cada corpus) pode ser visualizada na Figura 54.[45].

Figura 54 - Comparação de pausas silenciosas entre os corpora.



Fonte: Elaborado pelo autor

Na Figura 54, a disposição da caixa laranja em posição mais elevada, com média (0,821 s, std = 0,773) e mediana (0.572 s) superiores às da caixa azul (média = 0,558, mediana = 0,48, std = 0,323) são argumentos para afirmar que as pausas

[45]Ocasionais outliers maiores do que 4 segundos foram omitidos para melhorar a visualização.

silenciosas dos pacientes tiveram maior duração do que as do não pacientes ($p < 0,01$[46]). Portanto, os pacientes com esquizofrenia analisados foram mais disfluentes do que pessoas sem essa condição nessa variável.

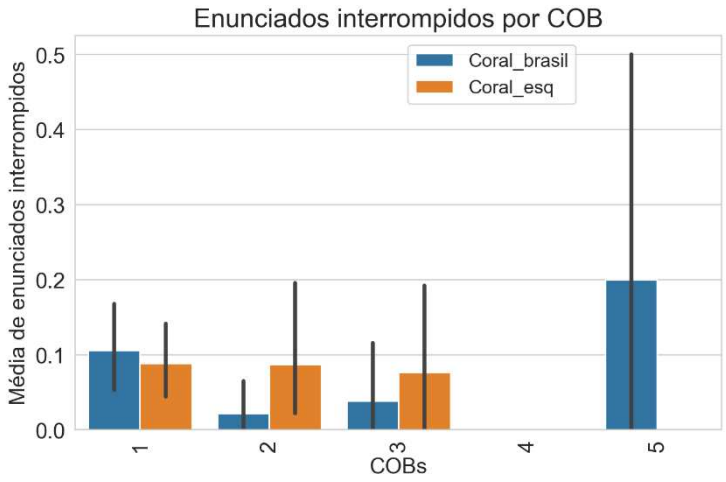
Em futuros trabalhos de viés mais sintático e prosódico, essas pausas silenciosas podem ser separadas por categorias e localização no enunciado, tais como pausas silenciosas após pausas preenchidas; após enunciados interrompidos; ou ainda pausas de respiração ou de acordo com sua fronteira sintática, como pausa na fronteira de sintagma ou início de uma oração. Dessa forma, será possível entender melhor a influência de elementos para além do escopo desta tese na duração de pausas silenciosas na fala desses pacientes.

5.2.3 Enunciados interrompidos por COB

Se considerados apenas a quantidade desse fenômeno nas amostras, a análise dos enunciados interrompidos é bastante inconclusiva porque há bastante oscilação dos resultados, os quais não têm relevância estatística.

Em stanzas com 1 COB, por exemplo (Figura 55), o grupo do C-ORAL-ESQ realizou menos enunciados interrompidos. Nas de 2 COBs, os pacientes realizaram 5 vezes mais desse fenômeno, além de quantidade também superior nas amostras de 3 COBs Stanzas com 4 COBs e 5 COBs não registraram esse tipo de disfluência no grupo do C-ORAL-ESQ.

Figura 55 – Comparação da frequência de enunciados interrompidos por COB.

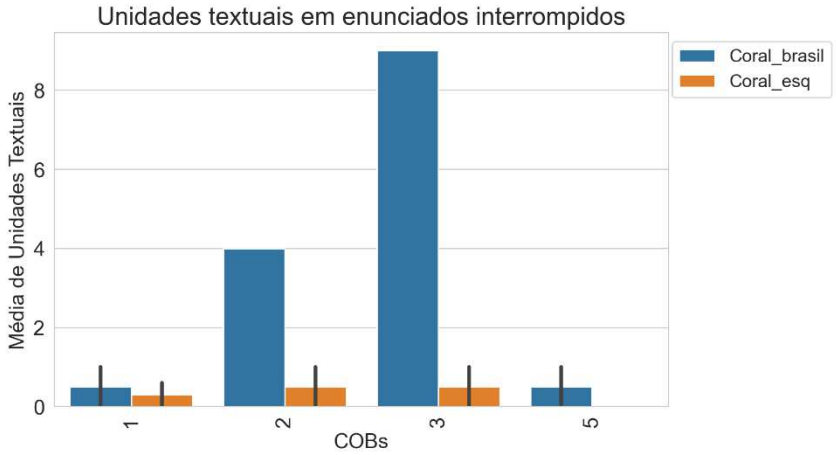


[46] Valor de p em notação científica = 9.459617899651488e-09

Fonte: Elaborado pelo autor.

Se considerada a elaboração textual desses enunciados interrompidos antes de sua interrupção, é possível afirmar que os pacientes elaboraram menos seus enunciados antes dessa interrupção em todas as amostras, conquanto isso não seja significativo.

Figura 56 - Frequência de unidades textuais em enunciados interrompidos.



Fonte: Elaborado pelo autor.

Em futuros trabalhos, é possível analisar a configuração interna desse tipo de disfluência e sua relação com o padrão informacional por meio de um novo balanceamento na ocasião de montar o grupo de pesquisa, o qual deve conter quantidades iguais ou ao menos semelhantes de enunciados interrompidos. Além disso, é possível analisar o motivo dessa interrupção com esse tipo de viés mais qualitativo.

5.2.4 Retractings e palavras retratadas

A quantidade de retractings (Figura 57) no grupo do C-ORAL-ESQ foi maior na amostra de 1, 2 e 5 COBs, e menor nas outras, mas sem significância estatística.

A quantidade de palavras retratadas (Figura 58) em cada retracting parece acompanhar a distribuição anterior, com a diferença de que os pacientes realizaram menos palavras retratadas na amostra de 4 COBs, na qual realizaram mais retractings. Há uma aparente maior quantidade de palavras retratadas em stanzas mais longas, como na de 5 COBs, mas nenhum resultado foi relevante.

Figura 57 – Comparação da frequência de retractings por COB.

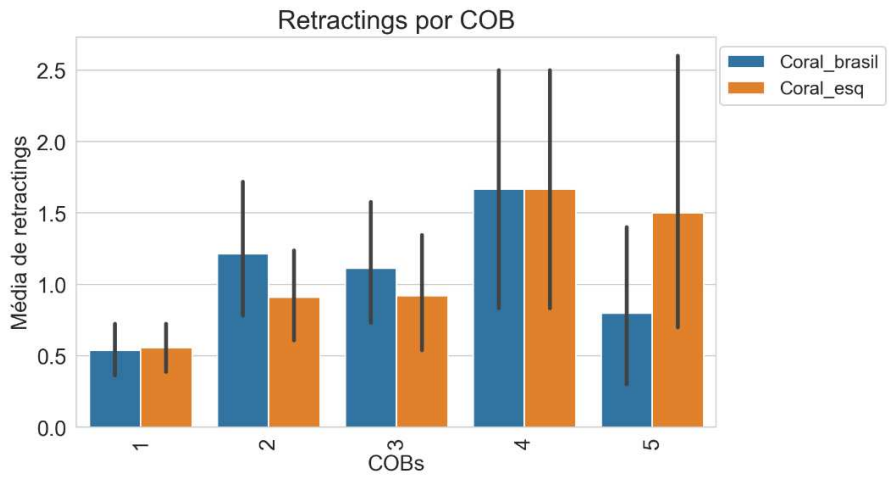
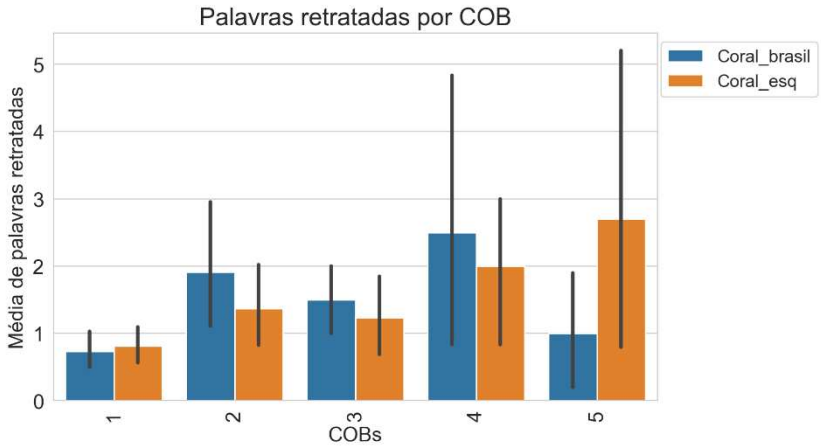


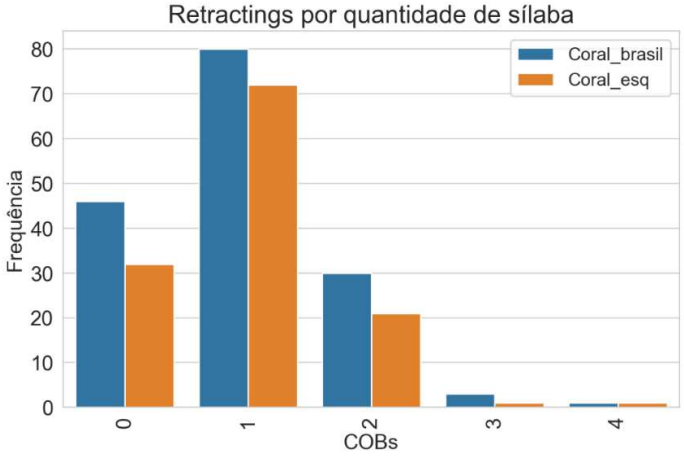
Figura 58 - Comparação da frequência de palavras retratadas por COB.



Fonte: Elaborado pelo autor.

Em geral, os pacientes realizaram essa disfluência com quantidade inferior de sílabas (Figura 59) do que o grupo do C-ORAL-ESQ.

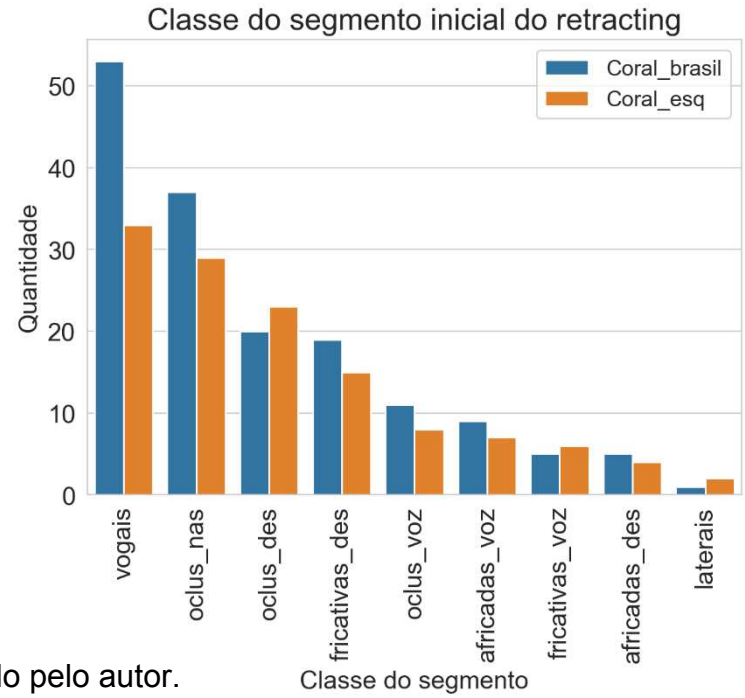
Figura 59 - Comparação da quantidade de sílabas dos retractings por COB.



Fonte: Elaborado pelo autor.

Em relação ao segmento inicial das palavras retratadas, é possível observar bastante similaridade na distribuição. É necessário ressaltar que essas amostras não estão balanceadas por quantidade de retractings, que são mais abundantes no C-ORAL-BRASIL. Apesar disso, é possível verificar que, em ambos os grupos, essas disfluências começam principalmente por vogais e oclusivas nasais, seguidas por fricativas desvozeadas (Figura 60). É possível observar que até mesmo as classes menos frequentes em um grupo também o são no outro. Esse tipo de dado pode sugerir que o estudo dos retractings desses pacientes talvez seja mais produtivo se considerados fatores menos estritamente fonéticos – principalmente os menos específicos - como a classe do segmento, que são basicamente as mesmas para os dois grupos.

Figura 60 – Comparação da classe fonética do primeiro segmento do retracting por corpus



Fonte: Elaborado pelo autor.

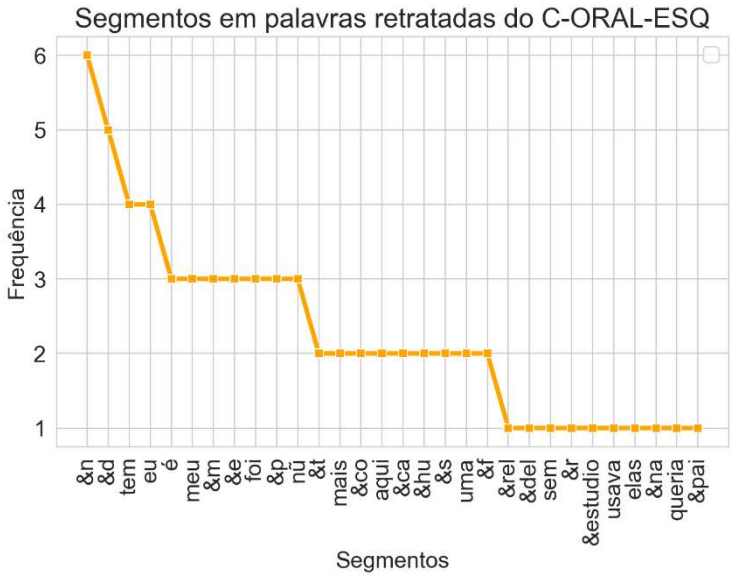
Nas Figuras 61 e 62, é possível visualizar os 30 segmentos mais frequentes nesse fenômeno nos dois grupos.

No grupo de pacientes, prevaleceram reformulações nos segmentos *&ne* e *&d*, na forma verbal *tem* no pronome pessoal *eu*.

No grupo do C-ORAL-BRASIL, se destacam os retractings em &f e *nu* (não), além da vogal *o*. Apesar de ser inconclusivo dado aos poucos segmentos disponíveis

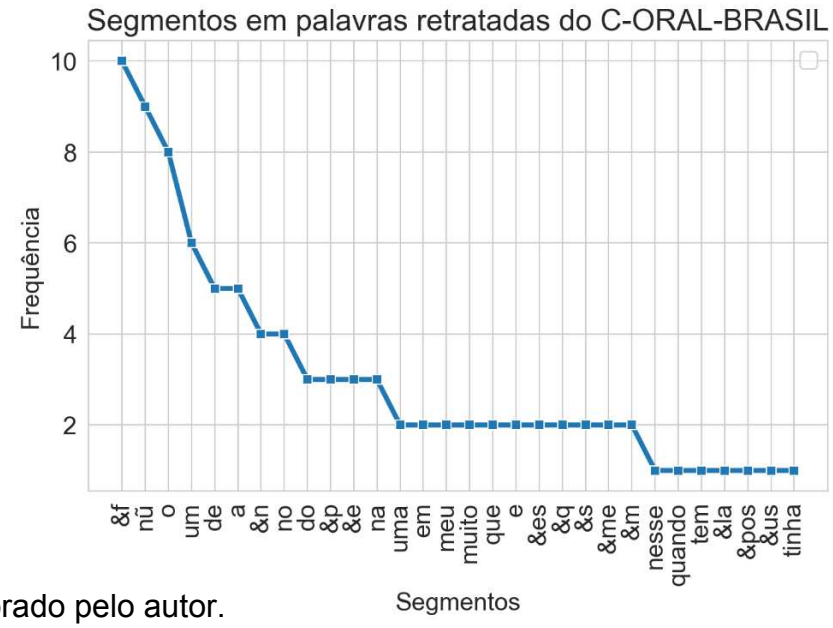
e a dificuldade de reconhecer todas as palavras, os segmentos mais frequentes de ambos os corpora parecem ser de termos mais gramaticais do que lexicais.

Figura 61 - Palavras retratadas mais frequentes no C-ORAL-ESQ



Fonte: Elaborado pelo autor.

Figura 62 - Palavras retratadas mais frequentes no C-ORAL-BRASIL



Fonte: Elaborado pelo autor.

5.3 Sumário de medidas da fala e disfluências

Pacientes com esquizofrenia realizam seus padrões informacionais em menor quantidade de unidades tonais na maioria das amostras. A quantidade de palavras de

suas stanzas é menor, assim como sua duração é geralmente menor. Suas palavras por segundo apresentaram resultado bastante similar ao do grupo do C-ORAL-BRASIL, exceto na amostra de 1 COB, na qual os pacientes foram mais rápidos nesse sentido e realizaram mais palavras por segundo.

Nessas comparações, pausas preenchidas e silenciosas apresentaram média e mediana superiores para os pacientes na comparação entre os dois grupos, mas apenas o segundo fenômeno foi relevante estatisticamente.

Já a análise de enunciados interrompidos mostrou que o grupo do C-ORAL-ESQ ora realiza menor quantidade desse fenômeno, na amostra mais relevante, de 1 COB, ora muito maior, como na de 2 ou 3 COBs. Em todas essas amostras, o grupo do C-ORAL-ESQ elaborou menos suas stanzas antes da interrupção.

Com relação aos retractings, foi possível observar que esses pacientes realizaram menor quantidade dessa disfluência do que os não pacientes nas amostras 1, 4 e 5 COBs. As palavras retratadas também seguem esse mesmo contorno de distribuição e ocorrem em maior quantidade no C-ORAL-BRASIL, exceto nas amostras de 1 e 5 COBs. Em ambos os grupos, esse fenômeno ocorre principalmente em palavras com 1 ou nenhuma sílaba, isto é, um segmento sem vogal. Classes fonéticas parecem ser insuficientes para distinguir essas disfluências, visto que ambos os grupos compartilham os segmentos iniciais mais frequentes, vogais e oclusivas vozeadas, e também os menos frequentes, que são africadas desvozeadas e laterais.

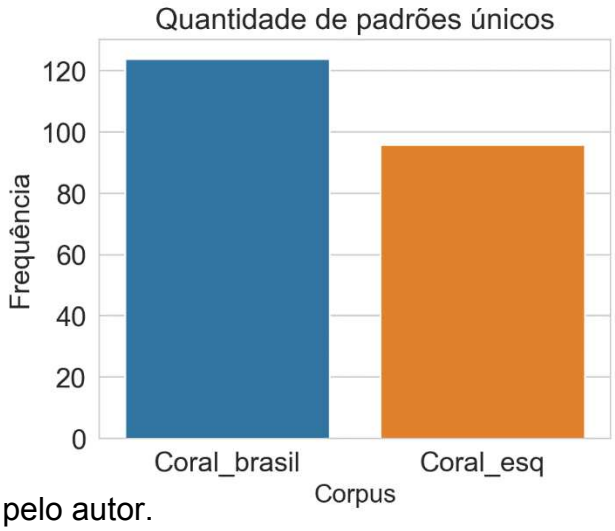
5.4 Comparação de padrões informacionais

Esta seção apresenta os resultados das comparações entre padrões informacionais e um comparativo individual entre unidades informacionais textuais, bem como uma comparação dos conjuntos de unidades dialógicas.

5.4.1 Quantidade de padrões únicos

Pacientes com esquizofrenia realizaram menor quantidade de padrões informacionais únicos (Figura 63) do que os participantes do C-ORAL-BRASIL (96 e 124, respectivamente).

Figura 63 - Quantidade de padrões únicos por corpus



Fonte: Elaborado pelo autor.

Em um primeiro momento, isso indica que pode haver menor variação da estrutura informacional no C-ORAL-ESQ, isto é, os padrões se repetem mais frequentemente neste grupo.

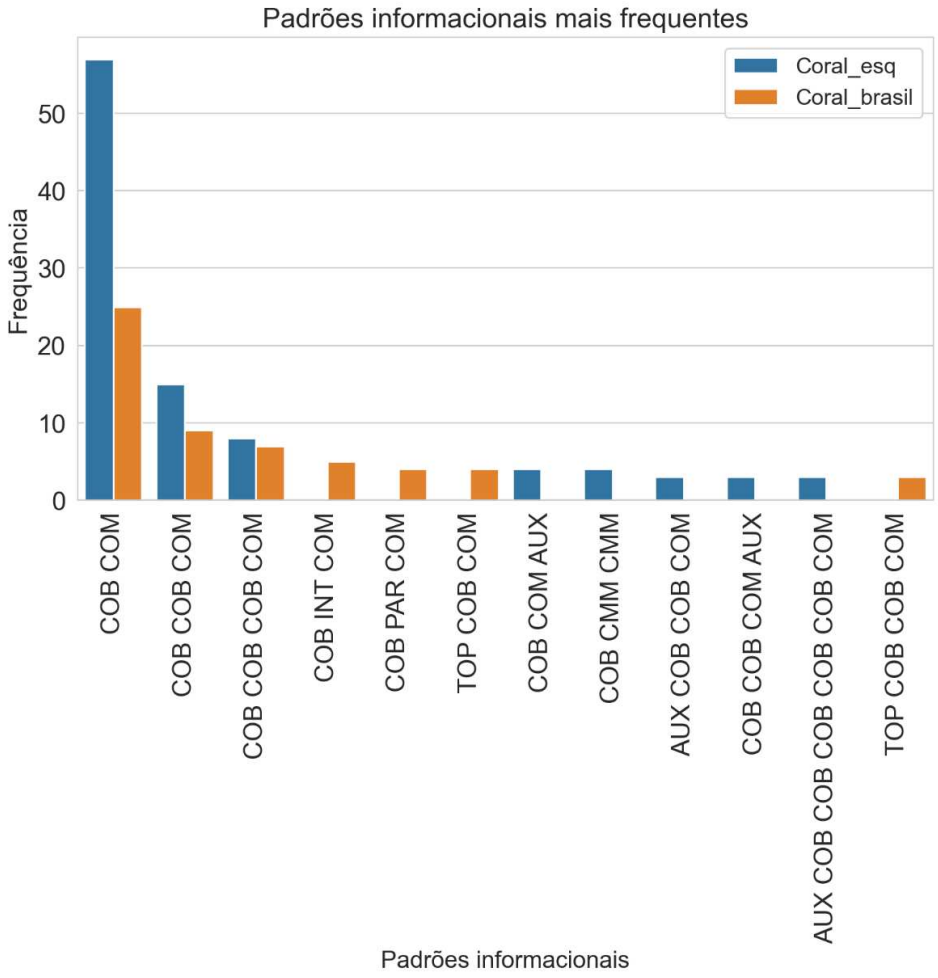
5.4.2 Padrões informacionais mais frequentes

Em ambos os grupos, o padrão mais frequente é o COB COM, com frequência quase três vezes superior no C-ORAL-ESQ (Figura 64).

É possível que o tipo de interação do C-ORAL-ESQ, que contem perguntas objetivas e mais curtas dirigidas ao paciente tanto no início quanto no final da interação, possa ter favorecido stanzas menores e menos elaboradas informacionalmente. No grupo do C-ORAL-BRASIL, bastante monológico, há poucas intervenções no monólogo do narrador e é esperado que haja estruturas mais complexas do COB COM.

Na Figura 64 também é possível observar que, em ambos os grupos, os 3 padrões mais frequentes apresentam apenas unidades textuais ilocucionárias. Posteriormente, os padrões do C-ORAL-BRASIL já começam a se complexificar, com Introdutor Locutivo, Parentético e Tópico, ao passo que o C-ORAL-ESQ não apresenta esses padrões.

Figura 64 - Padrões informacionais mais frequentes nos corpora.



Fonte: Elaborado pelo autor.

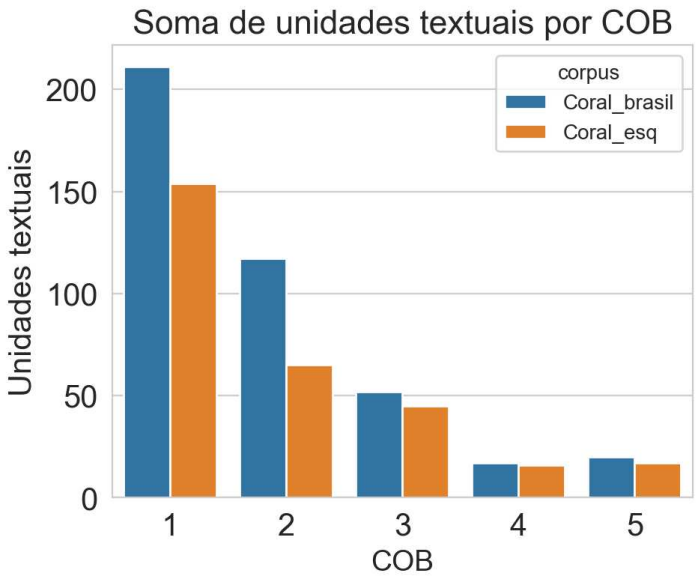
5.4.3 Unidades textuais totais

De forma esquemática, é possível observar que pacientes com esquizofrenia possuem menos unidades textuais totais por quantidade de COB (Figura 65) do que pessoas sem essa condição.

Em todas as amostras, independentemente do número de COBs, houve menor quantidade de unidades textuais totais. Até o momento, é possível ter uma inferência inicial de que, de fato, o padrão informacional de pessoas com esquizofrenia talvez seja menos elaborado do que pessoas sem essa doença mental.

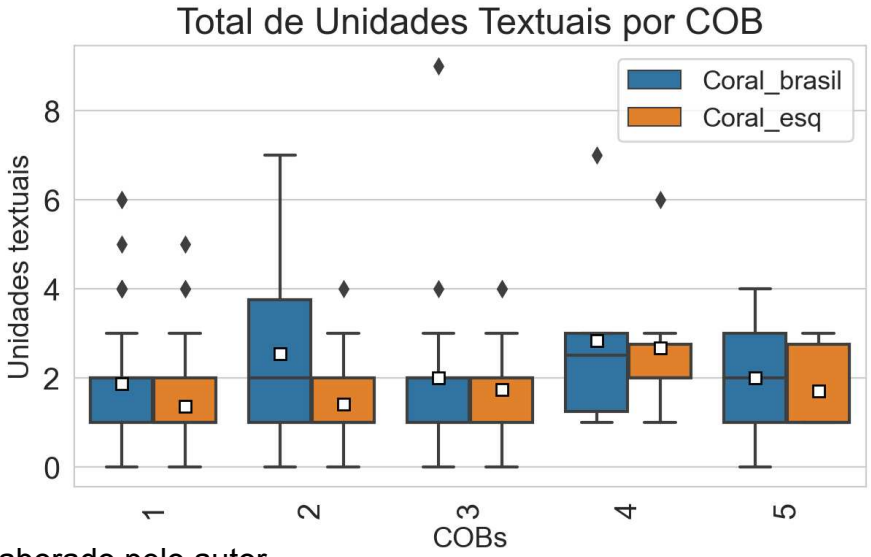
Isso é ainda mais realçado se se considera a Figura 66, que mostra a distribuição das unidades textuais por amostras.

Figura 65 - Comparação da soma de unidades textuais por COB.



Fonte: Elaborado pelo autor.

Figura 66 –Boxplot comparativo para total de unidades textuais por COB.



Fonte: Elaborado pelo autor.

Observa-se que a média de unidades textuais no grupo do C-ORAL-BRASIL é superior a média do grupo do C-ORAL-ESQ em todas as amostras, de 1 a 5 COBs. Apesar de haver alguns outliers em ambas as amostras, observa-se que até mesmo a mediana das amostras de 2, 4 e 5 COBs é superior no CORAL-BRASIL.

Há relevância estatística nessa diferença nas amostras de 1 e 2 COBs ($p = 0.0001$, $p = 0.001$, respectivamente, no teste U de Mann Whitney). Dado a relevância das amostras de 1 a 2 COBs, é possível sugerir que pacientes com esquizofrenia elaboram menos suas stanzas do ponto de vista informacional do que as pessoas sem

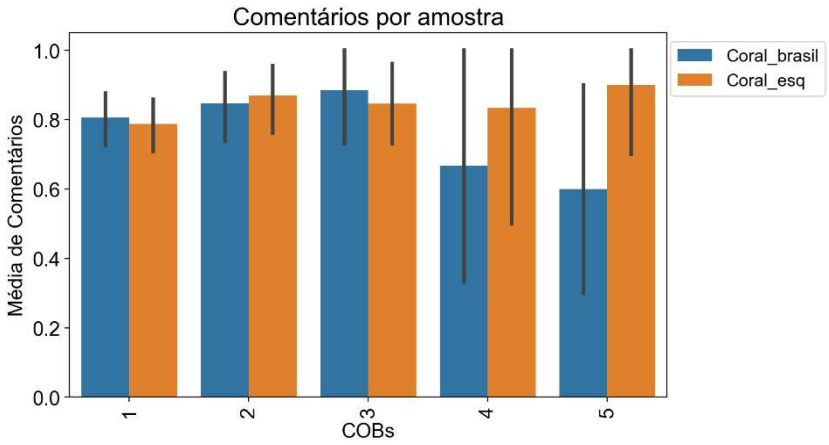
essa condição, pois há mais unidades textuais no grupo do C-ORAL-BRASIL do que no do C-ORAL-ESQ.

5.4.4 Unidades ilocucionárias

Unidade obrigatória de um enunciado ou de uma stanza, o Comentário aparentemente não serviria como base para comparar a riqueza ou não de uma unidade terminada, uma vez que a quantidade dessa unidade nos dois grupos seria, em tese, igual.

Ocorre que a quantidade de Comentários nas amostras não é equivalente nos dois grupos, e tampouco em amostras com mesmo número de COBs. Isso pode ser visto na Figura 67.

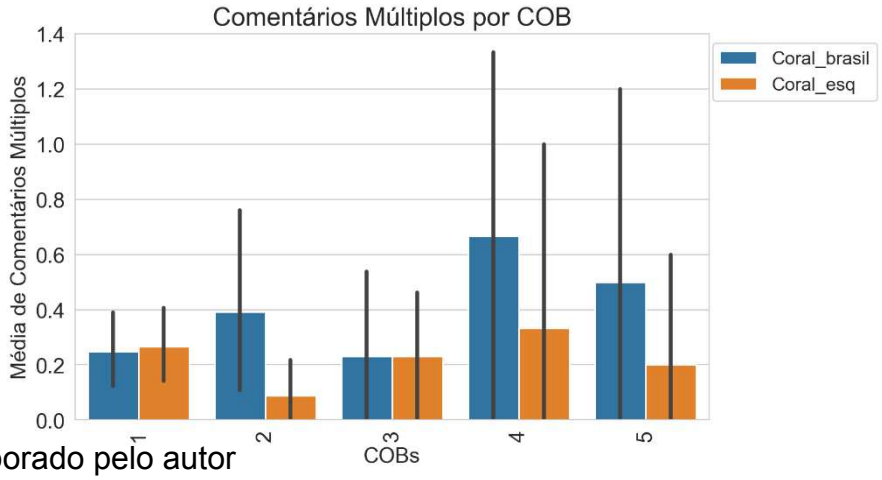
Figura 67 - Comparação de quantidade de Comentários por amostra.



Fonte: Elaborado pelo autor.

Aparentemente, a maior presença de Comentários Múltiplos nas stanzas do C-ORAL-BRASIL, além da destacada quantidade de enunciados interrompidos neste último grupo, são os motivos pelos quais existe essa diferença de quantidade de Comentários. Isso se deve ao fato de que stanzas que terminam com CMM não necessariamente precisam ter um Comentário, e enunciados interrompidos precisam ser interrompidos antes dessa unidade informacional para serem categorizados como tal. A média de ocorrência de CMMs pode ser vista na Figura 68 a seguir, assim como outras medições na Tabela 8.

Figura 68 - Comparação de quantidade de Comentários Múltiplos por COB.



Fonte: Elaborado pelo autor

Tabela 8 - Comentários Múltiplos por COB.

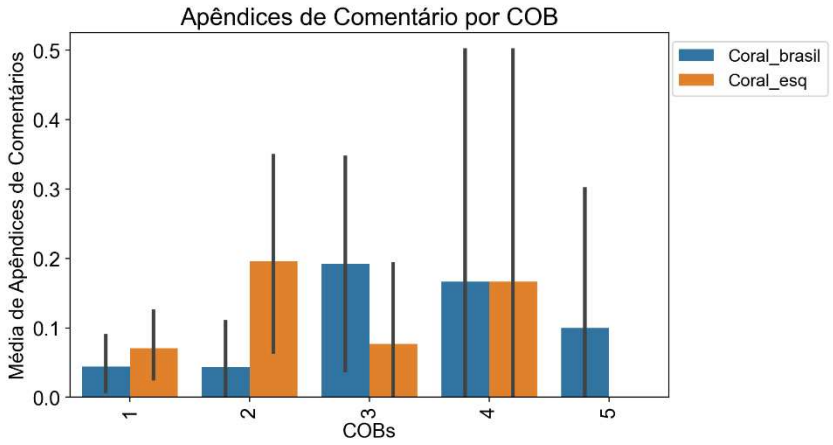
CMMs por COB						
corpus	COB	sum	max	mean	std	num_stanzas
Coral_brasil	1	28	3	0.25	0.74	113
Coral_brasil	2	18	6	0.39	1.14	46
Coral_brasil	3	6	2	0.23	0.65	26
Coral_brasil	4	4	2	0.67	1.03	6
Coral_brasil	5	5	3	0.5	1.08	10
Coral_esq	1	24	4	0.21	0.67	113
Coral_esq	2	2	2	0.04	0.29	46
Coral_esq	3	4	2	0.15	0.54	26
Coral_esq	4	2	2	0.33	0.82	6
Coral_esq	5	2	2	0.2	0.63	10

Fonte: Elaborado pelo autor.

De certa forma, quantidade superior de Comentários no C-ORAL-ESQ sugere que a ilocução neste corpus pode ser veiculada de forma menos variada, visto que há menos padrões de Comentários Múltiplos neste grupo de comparação. Em ambos os grupos, essa unidade foi mais frequente na amostra de 1 COB, mas bem menos abundante no C-ORAL-ESQ na de 2 COBs (soma de 4 no C-ORAL-ESQ e 18 no C-ORAL-BRASIL), com significância estatística (p=0,04).

Os Apêndices de Comentário (Figura 69), por sua vez, oscilaram em termos de frequência entre os dois grupos de comparação, mas, nas amostras mais relevantes, foram mais frequentes na fala de pacientes com esquizofrenia, mas sem significância estatística.

Figura 69 – Comparação de quantidade de Apêndices de Comentário por COB.



Fonte: Elaborado pelo autor.

É possível que isso reflita sua dificuldade de veicular a ilocução apenas na unidade de Comentário em stanzas mais curtas, alongando-as para a unidade tonal mais à direita. Na Tabela 9, por exemplo, é possível observar que esses pacientes realizaram até 2 Apêndices de Comentário em stanzas de 2 COBs. Ocorre que o desvio padrão nessa amostra é mais do que o dobro do C-ORAL-BRASIL, sinalizando que esses Apêndices de Comentário a mais podem ser meramente casuísticos, pois a maioria das stanzas teria apenas 1.

Tabela 9 - Apêndices de Comentário por COB.

Apêndices de Comentário por COB						
corpus	COB	sum	max	mean	std	num_stanzas
Coral_brasil	1	5	1	0.04	0.21	113
Coral_brasil	2	2	1	0.04	0.21	46
Coral_brasil	3	5	1	0.19	0.4	26
Coral_brasil	4	1	1	0.17	0.41	6
Coral_brasil	5	1	1	0.1	0.32	10
Coral_esq	1	8	1	0.07	0.26	113
Coral_esq	2	8	2	0.17	0.49	46
Coral_esq	3	1	1	0.04	0.2	26
Coral_esq	4	1	1	0.17	0.41	6
Coral_esq	5	0	0	0.0	0.0	10

Fonte: Elaborado pelo autor.

Em stanzas mais longas, os pacientes ora realizam menos Apêndices de Comentário, ora realizam quantidades iguais, como na amostra de 4 COBs, ora não

apresentam essa unidade informacional em suas stanzas, como na amostra de 5 COBs – esta última mais relevante do que a de 4 COBs.

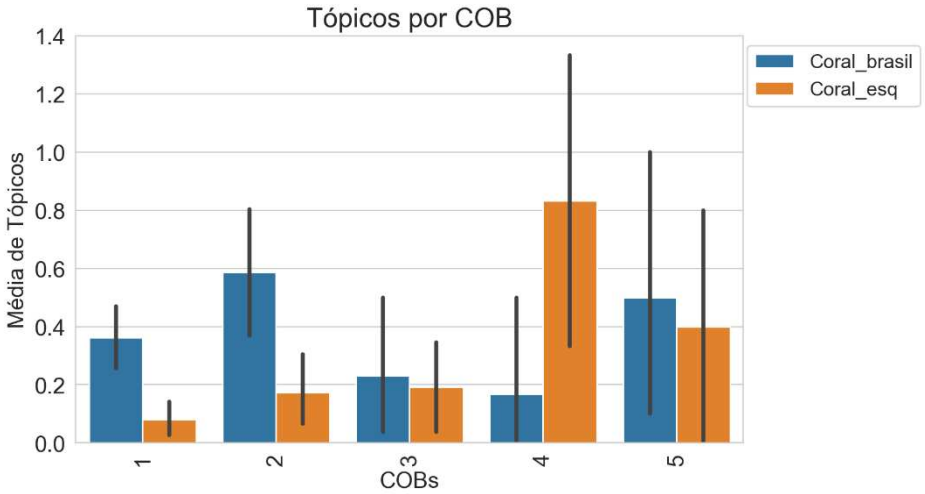
Aparentemente, a diferença entre unidades textuais ilocucionárias no C-ORAL-ESQ e C-ORAL-BRASIL se concentra prioritariamente nos Comentários Múltiplos, mais abundantes em todas as amostras de pessoas sem esquizofrenia. Em alguma medida, essa diferença entre os dois grupos também é visível nas stanzas mais curtas dos pacientes, uma vez que foram mais frequentes no C-ORAL-ESQ.

5.4.5 Tópicos e Apêndices de Tópico

Pacientes com esquizofrenia realizaram menos Tópicos do que sem essa condição nas amostras mais relevantes, de 1 a 2 COBs, com relevância estatística (p= 1.64025e-05 e p = 0.005, respectivamente). Isso pode ser visto na Figura 71.

Em stanzas de 3 COBs há similaridade na frequência dessa unidade entre os grupos, mas com quantidade máxima de Tópicos por stanza (ver Tabela 10) também predominante no grupo sem esquizofrenia. Na amostra de 4 COBs, o grupo do C-ORAL-ESQ realizou mais dessas unidades de forma destoante, mas trata-se de uma amostra com apenas 6 stanzas e sem significância estatística.

Figura 70 – Comparação da quantidade de Tópicos por COB.



Fonte: Elaborado pelo autor.

Naturalmente, esse resultado afeta os Apêndices de Tópico (Figura 71). Essa aparente diferença entre os dois corpora é desfeita pela Tabela 11, na qual é possível

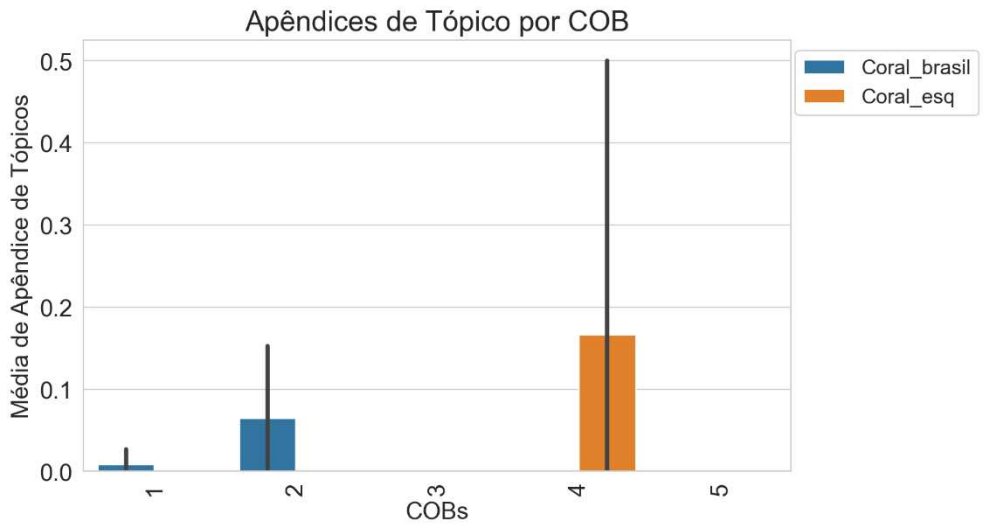
verificar que se trata apenas de 1 ocorrência não significativa isolada nas stanzas de 4 COBs do C-ORAL-ESQ

Tabela 10 – Quantidade de Tópicos por COB

Tópicos por COB						
COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	41	3	0.36	0.6	113
1	Coral_esq	11	2	0.1	0.35	113
2	Coral_brasil	27	3	0.59	0.78	46
2	Coral_esq	9	2	0.2	0.45	46
3	Coral_brasil	6	3	0.23	0.65	26
3	Coral_esq	6	1	0.23	0.43	26
4	Coral_brasil	1	1	0.17	0.41	6
4	Coral_esq	5	2	0.83	0.75	6
5	Coral_brasil	5	2	0.5	0.71	10
5	Coral_esq	4	2	0.4	0.7	10

Fonte: Elaborado pelo autor.

Figura 71 - Comparação da quantidade de Apêndices de Tópico



Fonte: Elaborado pelo autor.

Tabela 11 - Quantidade de Apêndices de Tópico por COB.

Apêndice de Tópico por COB						
COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	1	1	0.01	0.09	113
1	Coral_esq	0	0	0.0	0.0	113
2	Coral_brasil	3	1	0.07	0.25	46
2	Coral_esq	0	0	0.0	0.0	46
3	Coral_brasil	0	0	0.0	0.0	26
3	Coral_esq	0	0	0.0	0.0	26

4	Coral_brasil	0	0	0.0	0.0	6
4	Coral_esq	1	1	0.17	0.41	6
5	Coral_brasil	0	0	0.0	0.0	10
5	Coral_esq	0	0	0.0	0.0	10

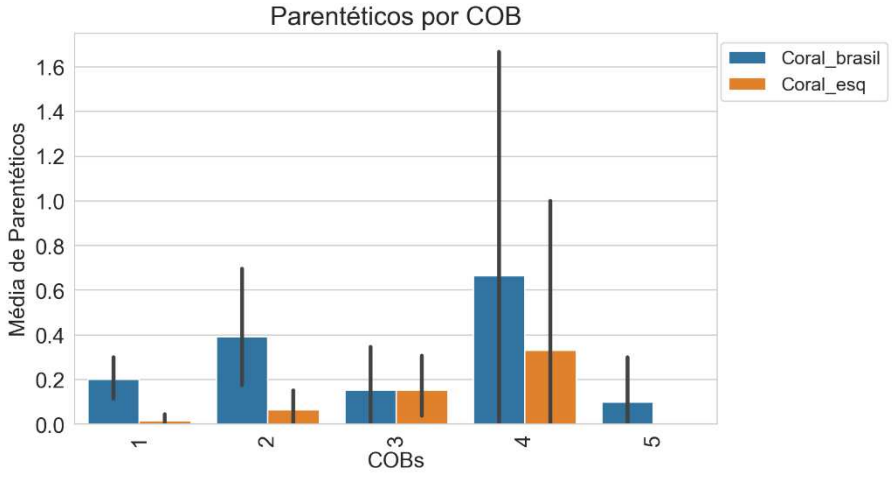
Fonte:Elaborado pelo autor.

Como o Tópico tem por característica prosódica grande variação da frequência fundamental em sua realização, sua baixa frequência nas stanzas dos pacientes pode favorecer uma fala menos variada em unidades textuais e , consequentemente, mais monótona do ponto de vista melódico. Além disso, a baixa quantidade dessa unidade textual em stanzas de até 2 COBs pode sugerir que os pacientes tenham dificuldade para sua realização e prefiram stanzas menos elaboradas textualmente.

5.4.6 Parentéticos

Com exceção da amostra com 3 COBs, pacientes com esquizofrenia realizaram menos Parentéticos na fala do que os sem esquizofrenia, com relevância estatística na amostra de 1 COB (p = 0.03 no teste U de Mann Whitney).

Figura 72 – Comparação da quantidade de Parentéticos por COB.



Fonte: Elaborado pelo autor.

A menor presença dessas unidades na amostra de 1 COB do grupo do C-ORAL-ESQ poderia sugerir que os pacientes inserem menos Parentéticos – como sequências explicativas – em suas stanzas mais curtas do que os participantes do C-ORAL-BRASIL, por exemplo.

Aparentemente, fenômenos como perda de foco, isto é, quando o falante insere comentários excessivos sobre um assunto não principal na interação, são bastante plausíveis de ocorrerem na unidade de Parentético, uma vez que esta é utilizada como um tipo de comentário acerca de outra unidade informacional. Sua ocorrência mais abundante no C-ORAL-BRASIL, entretanto, requisita uma abordagem mais individualizada das stanzas em questão para avaliar se esse fenômeno ocorre mais ou menos na fala dos pacientes do C-ORAL-ESQ do que na dos participantes do C-ORAL-BRASIL.

Tabela 12 - Quantidade de Parentéticos por COB.

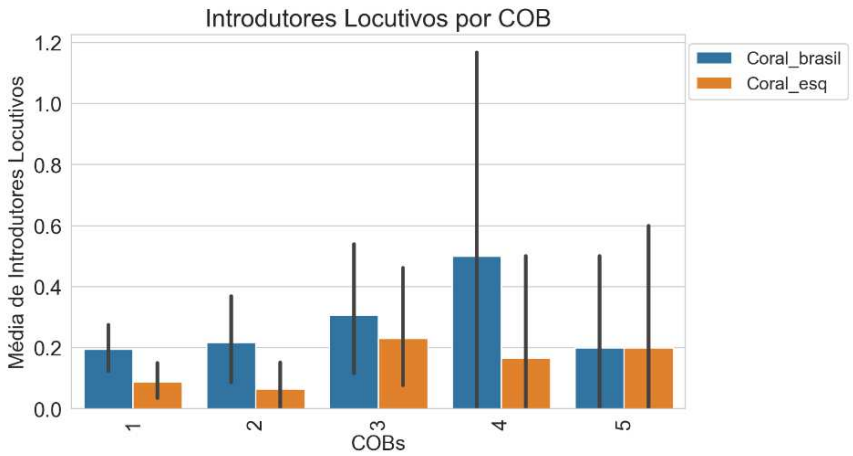
Parentéticos por COB						
COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	23	3	0.2	0.52	113
1	Coral_esq	8	1	0.07	0.26	113
2	Coral_brasil	18	4	0.39	0.91	46
2	Coral_esq	4	1	0.09	0.28	46
3	Coral_brasil	4	2	0.15	0.46	26
3	Coral_esq	5	1	0.19	0.4	26
4	Coral_brasil	4	3	0.67	1.21	6
4	Coral_esq	2	2	0.33	0.82	6
5	Coral_brasil	1	1	0.1	0.32	10
5	Coral_esq	0	0	0.0	0.0	10

Fonte: Elaborado pelo autor.

5.4.7 Introdutores Locutivos

Pacientes com esquizofrenia realizaram menor quantidade de Introdutores Locutivos em todas as amostras, com relevância estatística na amostra mais significativa, de 1 COB (p= 0,01 no teste U de Mann Whitney). A frequência de ocorrência dessa unidade textual pode ser visualizada na Figura 73 e na Tabela 13.

Figura 73 – Comparação da quantidade de Introdutores Locutivos por COB.



Fonte: Elaborado pelo autor.

No grupo do C-ORAL-ESQ, é possível que esta unidade informacional seja mais frequentemente utilizada em stanzas maiores, como as que os pacientes contam histórias ou falam de si ou das vozes que por ventura ouvem. Em geral, isso ocorre depois de perguntas menos relacionadas a assuntos de rotina e cumprimentos do início da consulta, que, usualmente, requisitam padrões informacionais menos complexos e que tradicionalmente não demandam suspensão pragmática e inserção de discurso reportado, este último um grande indicador de Introdutor Locutivo.

Introdutores Locutivos por COB

COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	22	1	0.19	0.4	113
1	Coral_esq	10	2	0.09	0.32	113
2	Coral_brasil	10	2	0.22	0.51	46
2	Coral_esq	3	1	0.07	0.25	46
3	Coral_brasil	8	2	0.31	0.55	26
3	Coral_esq	6	1	0.23	0.43	26
4	Coral_brasil	3	2	0.5	0.84	6
4	Coral_esq	0	0	0.0	0.0	6
5	Coral_brasil	2	1	0.2	0.42	10
5	Coral_esq	2	2	0.2	0.63	10

Fonte: Elaborado pelo autor.

Tabela 13 – Quantidade de Introdutores Locutivos por COB.

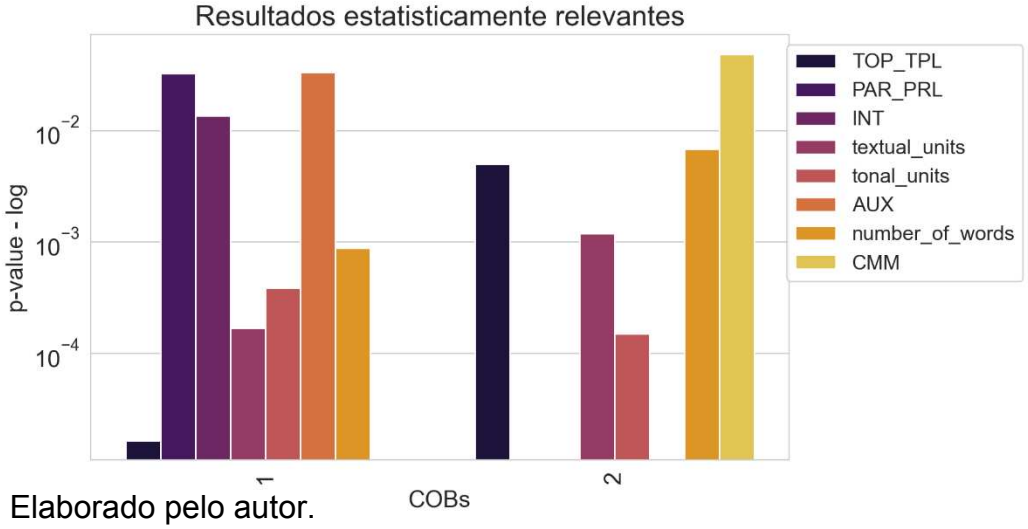
Nesse sentido, a tipologia extremamente monológica do C-ORAL-BRASIL pode favorecer suas stanzas de 1 COB. Mais comprometidos com contar uma história do que responder a perguntas de rotina, como os pacientes usualmente fazem no começo e fim das consultas, o grupo do C-ORAL-BRASIL tem a oportunidade de elaborar suas stanzas de forma mais complexas desde o início, e isso inclui realizar o Introdutor Locutivo para inserir outras vozes que enriqueçam suas narrativas.

5.5 Sumário das discussões sobre unidades textuais

Pacientes com esquizofrenia realizam stanzas com menor quantidade de unidades textuais totais na maioria das amostras, com relevância estatística. Em suas unidades ilocucionárias, se destaca o baixo número de Comentários Múltiplos. Há uma quantidade particularmente baixa de Tópicos em suas stanzas nas amostras

mais relevantes, de 1 a 2 COBs. Já os Parentéticos foram mais frequentes no C-ORAL-BRASIL do que no C-ORAL-ESQ, exceto em amostras de 3 COBs. Por fim, os introdutores locutivos foram menos frequentes na fala dos pacientes do que na dos pacientes, principalmente na amostra mais significativa. Como existe uma correspondência entre estrutura informacional da fala e sua melodia, é possível que a menor ocorrência de unidades textuais nas stanzas torne a fala dos pacientes menos articulada do ponto de vista informacional e mais monótona do ponto de vista prosódico. Os resultados estatisticamente relevantes encontrados nas comparações entre padrões informacionais estão plotados na Figura 74.

Figura 74 – Resultados estatisticamente significativos por quantidade de COB nas comparações entre padrões informacionais.



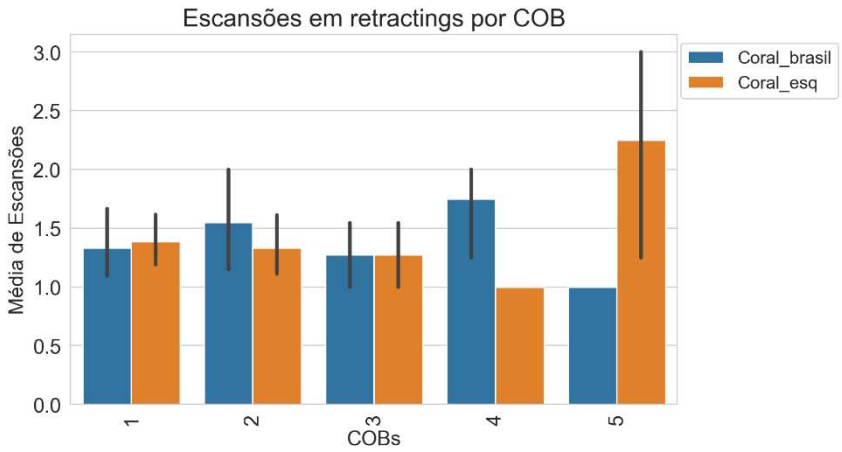
Fonte: Elaborado pelo autor.

5.6 Escansões dentro e fora de retractings

Neste trabalho, as Escansões de retracting são diferenciadas das Escansões de unidades informacionais sem essa disfluência. Isso se justifica devido ao fato de que escansões em retractings necessariamente ocorrem após uma disfluência e sinalizam que nem todas as palavras da unidade anterior ao retracting foram reformuladas, enquanto escansões de unidades informacionais não necessariamente indicam uma disfluência, pelo menos não as registradas nas transcrições, tais como pausas silenciosas. Portanto, quantificar esses dois tipos de Escansão separadamente pode dar pistas se os pacientes escandem mais ou menos as unidades textuais com e sem disfluências.

No que diz respeito às Encansões em retractings,, que podem ser visualizadas tanto na Figura 75 quanto na Tabela 14, foram registradas frequências muito similares em amostras até 3 COBs. Nas outras amostras, pacientes realizaram menos Escansões nas de 4 COBs e quantidade bastante superior na de 5 COBs. Entretanto, não há relevância estatística nesses resultados. Na prática, isso significa que os pacientes escandiram unidades textuais com quantidade bastante similar de disfluências do que os não pacientes.

Figura 75 – Comparação de Escansões em retractings por COB.



Fonte: Elaborado pelo autor.

Tabela 14 – Quantidade de Escansões em retractings.

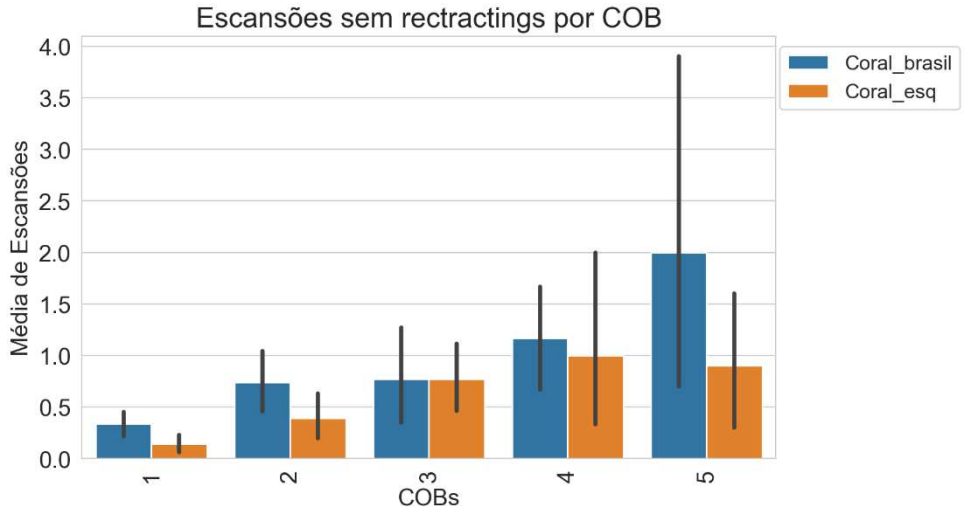
Escansões em retractings						
COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	28	4	1.33	0.73	113
1	Coral_esq	35	3	1.35	0.63	113
2	Coral_brasil	31	5	1.55	1.05	46
2	Coral_esq	20	3	1.43	0.65	46
3	Coral_brasil	14	2	1.27	0.47	26
3	Coral_esq	11	2	1.22	0.44	26
4	Coral_brasil	7	2	1.75	0.5	6
4	Coral_esq	2	1	1.0	0.0	6
5	Coral_brasil	2	1	1.0	0.0	10
5	Coral_esq	9	3	2.25	0.96	10

Fonte: Elaborado pelo autor.

Já as Escansões sem retractings se relacionam diretamente à quantidade de unidades textuais, pois a Escansão é a própria realização da unidade textual em

questão, mas ocupando mais de uma unidade tonal. Diferentemente das Escansões com retractings, esse tipo de Escansão não necessariamente se relaciona à alguma disfluência da fala, mas pode indicar pausas silenciosas nelas presentes, ou algum tipo de dificuldade em realizar uma unidade textual em apenas uma unidade tonal. As Escansões sem retractings desta comparação são plotadas na Figura 76.

Figura 76 – Comparação da quantidade de Escansões sem retractings por COB.



Fonte: Elaborado pelo autor.

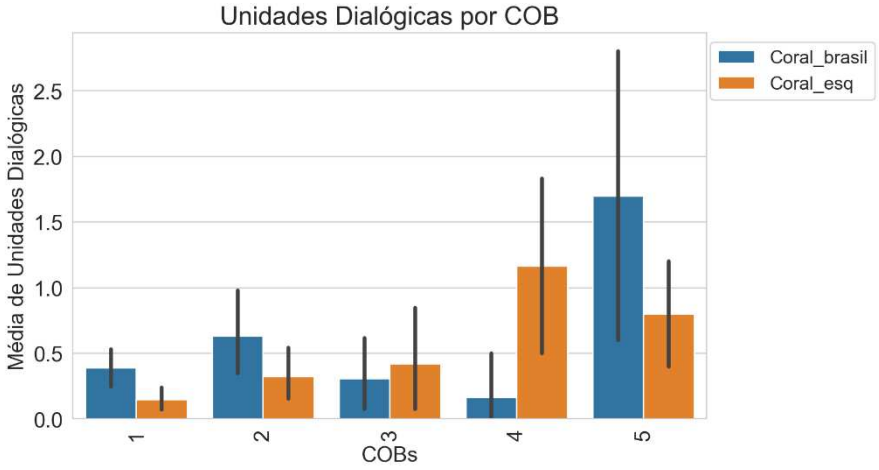
Conforme já foi discutido na seção 5.4.3, as stanzas do C-ORAL-ESQ têm menor quantidade de unidades textuais, de modo que seria natural, em um primeiro momento, que as Escansões sem rectractings também fossem menos frequentes. Isso é confirmado nas amostras mais relevantes, de 1 e 2 COBs, isto é, pacientes com esquizofrenia escandiram menos as unidades textuais em stanzas de 1 e de 2 COBs, com relevância estatística na de 1 COB (p = 0,03 no teste U de Mann Whitney). Isso significa que, além de terem menos unidades textuais, os pacientes as escandem menos à esquerda – e não necessariamente por uma disfluência na fala, visto que os retractings são eliminados nessa filtragem, conquanto haja pausas silenciosas, não marcadas na transcrição.

5.7 Unidades dialógicas

Pacientes com esquizofrenia realizaram menos unidades dialógicas em stanzas de 1, 2 e 5 COBs, e maior quantidade dessas unidades em stanzas com 3 e 4 COBs (cf. Figura 77 e Tabela 15) com relevância estatística nas de 1 COB (p = 0.03).

Na prática, isso significa que os pacientes apresentaram fala menos interativa nessa amostra, isto é, menos dirigida ao interlocutor de alguma forma, apesar de o ambiente de consulta médica aparentemente requisitar essa estruturação da fala, já que eles precisam interagir com seus médicos. Já a grande interatividade nas stanzas do C-ORAL-BRASIL nas amostras de 1 COB pode sugerir o interesse do narrador em manter seu turno de fala ou chamar a atenção para algum ponto de sua história.

Figura 77 – Comparação de unidades dialógicas totais por COB.



Fonte: Elaborado pelo autor.

Tabela 15 – Quantidade de unidades dialógicas totais por COB.

Unidades dialógicas por COB						
COB	corpus	sum	max	mean	std	num_stanzas
1	Coral_brasil	44	3	0.39	0.81	113
1	Coral_esq	16	2	0.14	0.44	113
2	Coral_brasil	29	4	0.63	1.1	46
2	Coral_esq	17	2	0.37	0.64	46
3	Coral_brasil	8	2	0.31	0.74	26
3	Coral_esq	12	4	0.46	0.99	26
4	Coral_brasil	1	1	0.17	0.41	6
4	Coral_esq	8	2	1.33	1.03	6
5	Coral_brasil	17	5	1.7	2.0	10
5	Coral_esq	8	2	0.8	0.79	10

Fonte: Elaborado pelo autor.

6 CONSIDERAÇÕES FINAIS

O grupo de pacientes do C-ORAL-ESQ realiza suas stanzas em menor quantidade de unidades tonais na maioria das amostras, com relevância estatística nas de 1 e 2 COBs ($p < 0,0001$). Essas stanzas têm menor duração do que as dos não pacientes nas amostras mais relevantes, exceto na de 3 COBs. A quantidade de palavras dessas realizações é menor no grupo do C-0RAL-ESQ em stanzas de 1 e 2 COBs ($p = 0,0008$ e $p = 0,006$), e os pacientes têm quantidades similares de palavras por segundo do que não pacientes, exceto na amostra de 1 COB, na qual o grupo do C-ORAL-ESQ foi mais rápido nesse sentido.

As pausas dos pacientes tiveram maior duração, tanto as preenchidas quanto as silenciosas. Na comparação de 76 pausas preenchidas (n=38 por grupo), o grupo do C-ORAL-ESQ apresentou média e mediana com maior duração do que os não pacientes, mas sem significância estatística ($p = 0,232$). Nas silenciosas (n = 600 por grupo), foi constatado que os participantes com esquizofrenia realizaram pausas também com maior média e mediana, com relevância estatística ($p < 0,01$).

Com relação a enunciados interrompidos, foi constatado que pacientes realizaram menos desse fenômeno em stanzas com 1 COB e mais na amostra de 2 COBs. Esses enunciados apresentaram maior quantidade de unidades textuais no grupo do C-ORAL-BRASIL do que no C-ORAL-ESQ em todas as amostras. Apesar de inconclusivo e sem relevância estatística, pois é necessário comparar a mesma quantidade de enunciados interrompidos em ambos os grupos para ter-se uma medida mais precisa, os resultados podem sugerir que indivíduos com esquizofrenia elaboram menos suas stanzas do ponto de vista informacional, pois realizam esses enunciados com menos unidades textuais, aparentemente.

Os retractings, por sua vez, foram mais comuns no C-ORAL-ESQ nas amostras de 1, 4 e 5 COBs. As palavras retratadas seguem distribuição semelhante, exceto na amostra de 5 COBs, na qual os pacientes parecem ter realizado mais palavras retradas do que os não pacientes de forma destoante, mas sem relevância estatística. Nos dois grupos, esse fenômeno tinha, em sua maioria, 1 ou nenhuma sílaba, isto é, um segmento sem vogal. As classes fonéticas dos segmentos iniciais dessas reformulações foram, em geral, bastante compartilhadas em termos proporcionais de frequência, tanto as mais frequentes, que foram as vogais e oclusivas nasais, quanto as menos frequentes, que foram as laterais. Isso pode sugerir que pesquisas em torno

desse tipo de disfluência talvez sejam mais produtivas se considerados fatores prosódicos como qualidade acústica e duração.

O padrão informacional mais frequente em ambos os corpora é o COB COM, com frequência muito superior no C-ORAL-ESQ em igual quantidade de stanzas. Por um lado, isso pode sugerir que os pacientes elaboram pouco suas stanzas a ponto de enriquecê-las menos com outras unidades textuais, como de fato ocorreu, conforme já discutido. Por outro, isso também pode refletir a natureza da interação do C-ORAL-ESQ, que é muito mais dialógica do que o C-ORAL-BRASIL e requisita que os pacientes respondam perguntas em geral muito curtas, principalmente no início e no final da consulta. No minicorpus do C-ORAL-BRASIL utilizado, de tipologia fortemente monológica, o narrador está muito mais engajado em contar suas histórias desde o início e, consequentemente, a requisição por padrões informacionais mais elaborados é mais latente.

Foram registradas diferenças marcantes se consideradas as unidades textuais em stanzas com igual quantidade de COBs, tal como proposto neste trabalho. Isso ocorre tanto quando se considera o total de unidades textuais por COB quanto essas unidades individualmente.

A média da soma de unidades textuais totais por COB é menor em pacientes do que em não pacientes em amostras de 1 ($p = 0{,}0001$), 2 ($p = 0{,}001$) COBs. Isso significa que, quando comparadas em igual quantidade de COBs, as stanzas dos pacientes são menos elaboradas textualmente, com consequências informacionais e melódicas para sua fala.

A comparação de unidade textual para unidade textual é ainda mais reveladora. A diferença já desponta nas unidades ilocucionárias obrigatórias na stanza. A despeito da igual quantidade de COBs, há menor quantidade de Comentários, provavelmente devido à frequência bastante inferior de Comentários Múltiplos – CMM - na fala dos pacientes. A respeito desse último, a diferença entre um grupo e outro foi estatisticamente relevante na amostra com 2 COBs ($p = 0{,}04$). Consequentemente, as stanzas dos pacientes possuem menor quantidade de padrões melódicos ilocucionários estabelecidos pela língua – função típica dos CMMs. Já os Apêndices de Comentário foram, em geral, mais frequentes no C-ORAL-ESQ, como nas amostras de 1 e 2 COBs, mas sem relevância estatística. É possível que esses pacientes tenham dificuldade para realizar stanzas curtas apenas na unidade de Comentário, estendendo-a mais à direita em forma de Apêndice.

No caso dos Tópicos, foram registradas frequências particularmente destoantes entre os grupos, com resultados estatisticamente relevantes nas amostras de 1 e 2 COBs. Devido à grande variação da frequência fundamental para sua realização, é possível que a baixa quantidade dessa unidade favoreça uma fala menos variada do ponto de vista informacional e mais monótona melodicamente.

Concernente aos Parentéticos, o grupo do C-ORAL-ESQ realizou menor quantidade dessa unidade nas amostras mais relevantes, com significância estatística na de 1 COB ($p = 0,03$). A hipótese de que os pacientes possam perder o foco nessa unidade não encontra subsídio nos dados analisados de forma quantitativa, como foi o caso desta tese, e é necessário investigar a ocorrência dessa unidade de forma mais qualitativa e individual.

Já os Introdutores Locutivos foram mais frequentes no C-ORAL-BRASIL do que no C-ORAL-ESQ, com diferença significativa na amostra de 1 COB ($p < 0,01$). É possível que os monólogos do C-ORAL-BRASIL favoreçam mais a tipologia narrativa e o discurso reportado, o qual é bastante realizado como Introdutor Locutivo. Como há muitos momentos que não são propriamente narrativos nas consultas psiquiátricas utilizadas neste trabalho, tal como momentos de cumprimentos iniciais entre médico-paciente, prescrição de remédios e assuntos de rotina; é possível que as stanzas do C-ORAL-ESQ favoreçam menos esse tipo de unidade textual do que o C-ORAL-BRASIL.

As escansões em retractings, que necessariamente dizem respeito a uma disfluência, tiveram distribuição semelhante entre os grupos, ao passo que as sem retracting, que indicam uma unidade textual escandida à esquerda, foram menos frequentes na fala dos pacientes nas amostras mais significativas, com resultado estatiscamente relevante na amostra de 1 COB ($p = 0,03$). Esse último resultado significa que os pacientes de fato escandiram menos suas unidades textuais em situações não necessariamente relacionadas à disfluência de reformulação da fala. Tal como já considerado, é esperado que, como o grupo do C-ORAL-ESQ realizou menor quantidade de unidades textuais, seria natural que também suas Escansões fossem menos frequentes.

Por fim, foi observado que as unidades dialógicas são menos frequentes nas stanzas de pacientes nas amostras de 1 ($p = 0,03$), 2 e 5 COBs. Suas stanzas seriam, então, menos interativas principalmente nas stanzas mais curtas, ainda que,

aparentemente, uma consulta psiquiátrica possa pressupor maior grau de interação e presença desse tipo de unidade, como para chamar a atenção do interlocutor.

Dito isso, é possível sugerir que pacientes com esquizofrenia realizam menos unidades textuais em suas stanzas do que pessoas sem essa condição nos dados analisados. Como existe uma contraparte prosódica de unidades informacionais, textuais ou não, isso significa que essa ausência também é refletida em sua fala em termos melódicos. Dessa forma, há menor variação de parâmetros acústicos típicos de unidades como Tópicos, que envolvem grande variação da frequência próximo à ilocução e são menos frequentes; de Parentéticos, que prosodicamente são regiões mais planas em termos de frequência e são menos recorrentes; de Introdutores Locutivos, que canonicamente disparam a taxa de elocução ao inserir, por exemplo, um discurso reportado. Se essas unidades ocorrem em escala menor e com relevância estatística, isso pode sugerir também que a fala desses pacientes poderia ser mais monótona porque sua estrutura informacional também o é, isto é, menos variada e abundante em unidades textuais.

REFERÊNCIAS BIBLIOGRÁFICAS

ADDINGTON, D.; ADDINGTON, J.; MATICKA-TYNDALE, E. **Assessing Depression Schizophrenia: The Calgary Depression Rating Scale**. **British Journal of Psychiatry** *163 (suppl. 22):*39-44, 1993.

ALPERT, M.; KOTSAFTIS, A.; POUGET, E. At Issue: speech fluency and schizoprenic negative signs. **Schizophrenia Bulletin**, v. 23, número 2, 1997, pp. 171-177.

ALPERT, M.; CLARK, A.; POUGET, E. The syntatic role of pauses in the speech of schizophrenic patients with alogia. **Journal of Abnormal Psychology,** v. 103, n° 4, 1994, pp. 750-757.

AUSTIN, J.L. **How to do things with words.** Oxford: Oxford University Press, 1962.

ANDREASEN, N. C. Positive and negative symptoms in schizophrenia. A critical reappraisal. **Archives of General Psychiatry** 1990, v. 47, pp. 615-621.

ANDREASEN, N. C.; Grove, W. M. Thought, language and communication in schizophrenia: Diagnosis and prognosis. **Schizophrenia Bulletin**, 12, 1986, pp. 356-359.

ANDREASEN, N. C. A scale for the assessment of thought, language and communication (TLC). **Schizophrenia Bulletin**, 12, 1986, pp. 473-482.

Artigos Originais • CoDAS 25 (4) • 2013 •

BEFI-LOPES, D. Complexidade da história e pausas silentes em crianças com e sem distúrbio específico de linguagem. **Revista CoDAS**. v.4, n° 25. 2013, pp. 325-329.

BIBER, D. Representativiness in Corpus Design. **Literary and Linguistic Computing,** v. 8, n° 4, 1993. Disponível em: http://otipl.philol.msu.ru/media/biber930.pdf

BIRD, S; KLEIN, E; LOPER, E. **Natural Language Processing with Python.** Analysing Text with the Natural Language Toolkit. Sebastospol, O'Reilly, 2009.

BLACK, D; GRANT, JON. **The essential companion to the diagnostic and statistical manual of mental disorders**. Fifth edition. Washington D.C; Londres; American Psychiatric Publishing, 2014.

CANÇADO, M.; AMARAL, L. **Introdução à Semântica Lexical.** Papéis temáticos, aspecto lexical e decomposição de predicados. Petrópolis, Editora Vozes, 2016.

CANÇADO, M.; AMARAL, L.; MEIRELLES, L. **VerboWeb: classificação sintático-semântica dos verbos do português brasileiro**. Banco de dados lexicais. UFMG. Disponível em: http://www.letras.ufmg.br/verboweb acesso em 24/01/2022.

CAVALCANTE, F. A. The topic unit in spontaneous American English: a corpus-based study. Dissertação (Mestrado). Faculdade de Letras. Programa de Pós-Graduação em Linguística, Universidade Federal de Minas Gerais.
Disponível em:
https://repositorio.ufmg.br/handle/1843/MGSS-A7GQ48 acesso em 17/03/2022

CAVALCANTE, F. A. The information unit of topic: a crosslinguistic, statistical study based on spontaneous speech corpora. Tese (Doutorado) – Faculdade de Letras. Programa de Pós-Graduação em Linguística, Universidade Federal de Minas Gerais, 2020.
Disponível em:
https://repositorio.ufmg.br/handle/1843/33673 acesso em 17/03/2022.

COMPTON, M.; LUNDEN, A.; CLEARY, S. et al. The aprosody of schizophrenia: Computationally derived acoustic phonetic underpinnings of monotone speech. **Schizophrenia Research**, n° 197, 2018, pp. 392–399.

CONDRAY, R.; STEINHAUER, S. R.; VAN KAMMEN, D.P.; KASPAREK, A. The language system in schizophrenia: Effects on capacity and linguistic structure. **Schizophrenia Bulletin**, 28, 2002, pp. 475-490.

COVINGTON, M. A. et al. (2005). Schizophrenia and the structure of language: The linguist's view. **Schizophrenia Research**, 77, 85-98.

CRESTI, E. **Corpus di Italiano parlato**. Firenze: Accademia della Crusca, 2000.

CRESTI, E.; DOVETTO, F. M.; ROCHA, B. Schizophrenia and prosody. First investigations. Em: MANFREDI, C. (Org.), **Models and analysis of vocal emissions for biomedical applications** - 9th international workshop - September 2-4, 2015. Firenze: Firenze University Press, 2015.

DOVETTO, F.M.; GEMELLI, M. **Il parlar matto. Schizofrenia tra fenomenologia e linguistica. Il corpus CIPPS**. Roma: Aracne, 2012.

ESPOSITO, Anna et al. The significance of empty speech pauses : cognitive and algorithmic issues. The Significance of Empty Speech Pauses: Cognitive and Algorithmic Issues. In: Mele F., Ramella G., Santillo S., Ventriglia F. (eds) **Advances in Brain, Vision, and Artificial Intelligence**. BVAI 2007. Lecture Notes in Computer Science, vol 4729. Springer, Berlin, Heidelberg, 2007, pp. 542-554.

FERREIRA-NETTO, W. *ExProsodia.* Revista da Propriedade Industrial – RPI, 2038, item 120, em 26 out. 2010.

HONNIBAL, M ; MONTANI, I**. SpaCy 2:** Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

HUNTER, J. D. Matplotlib: A 2D Graphics Environment, **Computing in Science & Engineering**, v. 9, n° 3, pp. 90-95, 2007.

JORGE, A. C. A. *Prosódia afetiva na esquizofrenia.* 2019. Dissertação (Mestrado) – Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, São Paulo, 2019.

JORGE, A. C. A. Prosódia afetiva na esquizofrenia. *Revista Estudos Linguísticos,* v.49, n.3, dezembro de 2020, pp. 1393-1412.

KAY, S.R; OPLER L.A.; LINDENMAYER J.P. The Positive and Negative Syndrome Scale (PANSS): Rationale and Standardisation. British Journal of Psychiatry (1989), 155 (suppl. 7), 59-65

KUPERBERG, G. Language in Schizophrenia. Part 1: An Introduction. **Language and Linguistics Compass** n° 4, v.8, 2010, pp. 576–589.

LENNES, Mieta. **Mark pauses** (2002, 2006).

Disponível em: http://phonetics.linguistics.ucla.edu/facilities/acoustic/mark_pauses.txt e em: https://lennes.github.io/spect/, acesso em 24/01/2022.

LIDDLE, P. F., NGAN, E. T. C., CAISSIE, S.L. et al. Thought and language index: An instrument for assessing thought and language in schizophrenia. **British Journal of Psychiatry**, n° 181, 2002, pp. 326-330.

MCKINNEY, W, et al. Data structures for statistical computing in python. In: **Proceedings of the 9th Python in Science Conference**. 2010. pp. 51–6.

MONEGLIA, M; RASO, T. Notes on Language into Act Theory (L-AcT). In: RASO, T.; MELLO, H. (Eds.). **Spoken corpora and linguistic studies**. Amsterdam: John Benjamins, 2014, p. 469–495.

MORICE, R. D., & Ingram, J. C. L. Language complexity and age of onset of schizophrenia. **Psychiatry Research**, 9, 1983, pp. 233-242.

MOTA, N.B.; COPELLI, M; RIBEIRO, S. Thought disorder measured as random speech structure classifies negative symptoms and Schizophrenia diagnosis 6 months in advance. **Schizophrenia.** v.3 n° 18. In: MOTA, N.B. **Mapeamento mental através da análise computacional do discurso**. Tese de Doutorado. Programa de Pós-Graduação em Neurociências. Universidade Federal do Rio Grande do Norte. 2017.

classifies negative symptoms and Schizophrenia diagnosis 6 months in advance.

MUNIZ, M. C. M. **Léxicos Computacionais: Desafios na Construção de um Léxico de Português Brasileiro**. Monografia de Qualificação. Instituto de Ciências Matemáticas de São Carlos, USP. 50p. Fev, 2003. Disponível em: http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/publicacoes.html, acesso em 24/01/2022.

MUNIZ, M. C. M**. A construção de recursos lingüístico-computacionais para o português do Brasil**: o projeto de Unitex-PB. Dissertação de Mestrado. Instituto de Ciências Matemáticas de São Carlos, USP. 72p. 2004. Disponível em: http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/publicacoes.html, acesso em 22/01/2022.

RASO, T.; MELLO, H. (Eds.). **C-ORAL-BRASIL I: Corpus de referência do português brasileiro falado informal**. 1. ed. Belo Horizonte: Editora UFMG, 2012.

ROCHA, B.N. **Uma metodologia empírica para a identificação e descrição de ilocuções e a sua aplicação para o estudo da Ordem em PB e Italiano**. Tese de doutorado. Faculdade de Letras, Universidade Federal de Minas Gerais. Disponível em: http://www.bibliotecadigital.ufmg.br/dspace/bitstream/handle/1843/MGSS-A8KGXK/tese___bruno_rocha.pdf?sequence=1 acesso em 02/02/2020

ROCHA, B.N. O corpus C-ORAL-ESQ e a estrutura informacional da fala de pacientes com esquizofrenia. **Working papers em Linguística,** 20 (1), pp. 212-238.

SALOMÃO, M. M. M. (2021). FrameNet Brasil: um trabalho em progresso. Calidoscópio, *7*(3), 171–182. Recuperado de http://revistas.unisinos.br/index.php/calidoscopio/article/view/4870, acesso em 24/01/2022.

STEUBER, L.C. **Disordered Thought, Disordered Language: A corpus-based description of the speech of individuals undergoing treatment for schizophrenia** (2011). Dissertations and Theses. Paper 63.

VAN ROSSUM, G.; DRAKE, J. **Python reference manual**. Centrum voor Wiskunde en Informatica Amsterdam; 1995.

WASKOM, M. et al, 2017. Mwaskom/seaborn: v0.8.1 (September 2017), Zenodo. Disponível em: https://doi.org/10.5281/zenodo.883859.

ANEXOS

Anexo A- Programa 1: Extrator de padrões informacionais e normalizador de ortografia[47]

Link para o programa (Google Colab): https://colab.research.google.com/drive/1A-gTv1ArN8HvFeunRfd088ornK9COyr9?usp=sharing

Link para atualizações:
https://github.com/carlosjuniorcosta1/extrator_1_coral_esq_e_brasil

Finalidade: Extração de padrões informacionais e medidas da fala, normalização ortográfica, etiquetagem por classes de palavras, distribuição de pausas preenchidas (se alinhadas), entre outros.
Tutorial em vídeo:
https://www.loom.com/share/15a2516728224fcc968cbeebc6093531

Entrada:
Transcrições alinhadas em formato xml[48], etiquetadas ou não.
Etiquetador de classes gramaticais treinado.
Opcional: pausas preenchidas das transcrições alinhadas separadamente. Nesse caso, a duração de toda a pausa preenchida deve estar alinhada – com fronteira à esquerda e à direita – e com a etiqueta =PAUSA_P=.

Saída:
Padrões informacionais e medidas específicas da fala de todos os participantes;
Medidas de tendência central e dispersão desses padrões;
Quantidade de pausas;
Quantidade de retractings e classificação de seus segmentos iniciais;
Normalização ortográfica da transcrição para português padrão;

[47]Todos os programas e suas versões atualizadas estão em meu Github: https://github.com/carlosjuniorcosta1
[48]O arquivo xml foi gerado no WinPitch. Adaptações para outros formatos são possíveis e fáceis de se serem feitas. Gentileza me contatar: carlosjuniorcosta1@gmail.com

Etiquetagem por classes gramaticais.
Subcorpus balanceado para comparações
Gráficos variados comparando os dois grupos do corpus balanceado
Testes estatísticos entre variáveis presentes nos dois grupos
Distribuição das pausas preenchidas de acordo com sua duração e também de acordo com a duração que essas pausas tiveram diante das classes gramaticais

```
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 23 23:25:55 2021
@author: José Carlos Costa
email: carlosjuniorcosta1@gmail.com
"""
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import re
import xml.etree.ElementTree as ET
from collections import Counter
import pickle
import nltk
from nltk import word_tokenize
import numpy as np
import os
from string import punctuation
nltk.download('punkt')
import xml.etree.ElementTree as ET
def limpa_coral(texto):
    import re
    texto = re.sub(r'(\*\w{3}\:)?(\[\d+\])?|\[?\/\d+\]?|\+|/|/{2}|=?i?\s?\-?\w{3}_?r?s?n?=\s?\$?', '', texto)
    texto = texto.replace('hhh', '').replace('yyyy', '')\
    .replace('yyy', '').replace('xxx', '').replace('<', '')\
    .replace('>','').replace('?', '').replace('=', '')
    texto = re.sub(r'&\w+', '', texto)
    texto = re.sub(r'\s+', ' ', texto)
    return texto
def limpa_coral_sobreposicao(texto):
    import re
    texto = re.sub(r'(\*\w{3}\:)?(\[\d+\])?|\[?\/\d+\]?|\+|/|/{2}|=?i?\s?\-?\w{3}_?r?s?n?=\s?\$?', '', texto)
```

```
    texto = texto.replace('hhh', '').replace('yyyy', '').replace('yyy', '').replace('xxx', '')\
    .replace('?', '').replace('=', '')
    texto = re.sub(r'&\w+', '', texto)
    texto = re.sub(r'\s+', ' ', texto)
    return texto
def normaliza_coral(texto):
    import re
    texto = re.sub(r'(\*\w{3}\:)?(\[\d+\])?|\[?\/\d+\]?|\+|/|/{2}|=?i?\s?\-?\w{3}_?r?s?n?=\s?\$?', '', texto)
    texto = texto.replace('hhh', '').replace('yyyy', '')\
    .replace('yyy', '').replace('xxx', '').replace('<', '')\
    .replace('>','').replace('?', '').replace('=', '')
    texto = re.sub('&\w+', '', texto)
    texto = re.sub(r'&\w+', '', texto)
    texto = re.sub(r'\s+', ' ', texto)
    texto = texto.replace("""", "'")
    texto = texto.strip()
    formas_conv ="""
```

ni (em), a’ (olha), acabamo (acabamos), achamo (achamos), agradecemo (agradecemos), a’ lá (olha), a’ (olha), a’ (olha), aprendemo (aprendemos), arrumamo (arrumamos), assinávamo (assinávamos), atravessamo (atravessamos), avi (vi), avinha (vinha), bebemo (bebemos), beijamo (beijamos), botemo (botamos), chegamo (chegamos), cheguemo (chegamos), choramo (choramos), colocamo (colocamos), começamo (começamos), comemo (comemos), comemoramo (comemoramos), compramo (compramos), conhecemo (conhecemos), conseguimo (conseguimos), contamo (contamos), conversamo (conversamos), corremo (corremos), cortamo (cortamos), deixamo (deixamos), descansamo (descansamos), descemo (descemos), devemo (devemos), empurramo (empurramos), encontramo (encontramos), entramo (entramos), envem (vem), envinha (vinha), escolhemo (escolhemos), esquecemo (esquecemos), estamo (estamos), estudemo (estudamos), evem (vem), falamo (falamos), fazido (feito), ficamo (ficamos), fize (fiz), fizemo (fizemos), fomo (fomos), for (formos), fraga (flagra), fragando (flagrando), frago (flagro), fumo (fomos), ganhamo (ganhamos), levamo (levamos), levantamo (levantamos), levantemo (levantamos), mandamo (mandamos), manti (mantive), o’ (olha), o’(olha), paramo (paramos), passamo (passamos), pedimo (pedimos), peguemo (pegamos), perdemo (perdemos), pinchando (pichando), pintemo (pintemos), podemo (podemos), precisamo (precisamos), pusemo (pusemos), resolvemo (resolvemos), saímo (saímos), seje (seja), sentamo (sentamos), sentemo (sentamos), separamo (separamos), somo (somos), sufro (sofro), temo (temos), tiramo (tiramos), tivemo (tivemos), tomamo (tomamos), trabalhamo (trabalhamos), trago (trazido), vesse (visse), viemo (viemos), vimo (vimos), tó (toma), cê (você), cês (vocês), e’ (ele), ea (ela), eas (elas), es (eles), ocê (você), ocês (vocês), aque’ (aquele), aquea (aquela), aqueas (aquelas), aques (aqueles), ca (com a), co (com o), cos (com os), cum (com um), cuma (com uma), d’ (de) d’(de), d’(de), d’(de), dum (de um), duma (de uma), dumas (de umas), duns (de uns), deerreí (na DRI), ni (em), num (em um), numa (em uma), numas (em umas), pa (para), pas (para as), p’(para), p’(para), p’(para),

p’(para), p’(para), p’(para), p’(para), po (para o), p’(para), pos (para os), p’(para), pra (para), pr’(para), pras (para as), pro (para o), pr’(para), pros (para os), prum (para um), pruma (para uma), pruns (para uns), p’(para), p’ (para), p’(para), pum (para um), puma (para uma), c’ aqueas (com aquelas), c’(com), c’ cê (com você), c’ e’ (com ele), c’ (com), c’(com essas), c’(com), c’ ocê (com você), c’ ocês (com vocês), daque’ (daquele), daquea (daquela), daqueas (daquelas), daques (daqueles), d’ cê (de você), de’ (dele), dea (dela), d’(de), d’(de), d’(de), des (deles), d’ es (de eles), d’(de), d’ ocê (de você), d’ ocês (de vocês), naque’ (naquele), naquea (naquela), naques (naqueles), ne’ (nele), n’ ocê (em você), n’ ocês (em vocês), p’ aque’ (para aquele), p’(para), p’ cê (para você), p’ cês (para vocês), p’ e’ (para ele), p’(para), p’ (para), p’(para), p’ es (para eles), p’ esse (para esse), p’ mim (para mim), p’ ocê (para você), p’ ocês (para vocês), pr’(para), pr’(para), pr’(para), pr’(para), pr’ ocê (para você), pr’ ocês (para vocês), p’ sio’ (para a senhora), p’ siora (para a senhora), p’(para), armoçar (almoçar), artinho (altinho), arto (alto), arto (alto), comprica (complica), compricar (complicar), cravícula (clavícula), escardada (escaldada), prano (plano), pranta (planta), pray (play), prissado (plissado), probremas (problemas), sortando (soltando), sortar (soltar), sortei (soltei), sorto (solto), sortou (soltou), vorta (volta), vortar (voltar), vortava (voltava), vorto (volto), nũ (não), canarim (canarinho), espim (espinho), padrim (padrinho), passarim (passarinho), porco-espim (porco-espinho), sozim (sozinho), almoçozim (almoçozinho), amarelim (amarelinho), azulzim (azulzinho), bebezim (bebezinho), bichim (bichinho), bocadim (bocadinho), bonitim (bonitinho), cachorrim (cachorrinho), cantim (cantinho), capoeirim (capoeirinhas), carrim (carrinho), cedezinho (CD), certim (certinho), certins (certinhos), Chapeuzim Vermelho (Chapeuzinho Vermelho), chazim (chazinho), controladim (controladinha), desfiadim (desfiadinho), direitim (direitinho), direitim (direitinho), esquisitim (esquisitinho), fechadim (fechadinho), filhotim (filhotinho), formulariozim (formulariozinho), fundim (fundinho), Geraldim (Geraldinho), golezim (golezinho), igualzim (igualzinho), instantim (instantinho), jeitim (jeitinho), Joãozim (Joãozinho), joguim (joguinho), ladim (ladinho), maciim (maciinho), mansim (mansinho), Marquim (Marquinho), meninim (menininho), morenim (moreninho), murim (murinho), Paulim (Paulinho), pequeninim (pequenininha), pertim (pertinho), negocim (negocinhos), partidim (partidinho), porquim (porquinho), portim (portinha), potim (potinho), pouquim (pouquinho), pozim (pozinho), pretim (pretinho), prontim (prontinho), quadradim (quadradinha), quadradim (quadradinho), queimadim (queimadinho), rapidim (rapidinho), recheadim (recheadinho), rolim (rolinho), tamanim (tamaninho), tampadim (tampadinho), terrenim (terreninho), tiquim (tiquinho), todim (todinho), toquim (toquinho), trancadim (trancadinhos), trenzim (trenzinho), tudim (tudinho), sio’ (senhora), sior (senhor), siora (senhora), sô (senhor), mó (muito), po’ (pode) ,tá (está) ,tamo (estamos) ,tamos (estamos) ,tão (estão) ,tar (estar) ,taria (estaria) ,tás (estás) ,tava (estava) ,tavam (estavam) ,távamos (estávamos) ,tavas (estavas) ,teja (esteja),teve (esteve) ,tive (estive) ,tiver (estiver) ,tiverem (estiverem) ,tivesse (estivesse) ,tô (estou) ,vamo (vamos) ,vão (vamos) ,vim (vir) ,xá (deixa), antiguim (antiguinho), banhozim (banhozinho), branquim (branquinho), certim (certinho),

devagarzim (devagarzinho), direitim (direitinho), direitim (direitinho), gostosim (gostosinho), limãozim (limãozinho), minutim (minutinho), pertim (pertinho), pulim (pulinho), pouquim (pouquinho), rapidim (rapidinho), recibim (recibinho), verdim (verdinho), xixizim (xixizinho), zerim (zerinho)

babacar (embabacar), zucrinando (azucrinando), zucrinar (azucrinar), brigado (obrigado), brigada (obrigada), baixa (abaixa), credita (acredita), creditei (acreditei), creditou (acreditou) , baixar (abaixar), baixei (abaixei), baulado (abaulado), bora (embora), borrecido (aborrecido), brigada (obrigada), brigado (obrigado), caba (acaba), cabar (acabar), cabava (acabava), cabei (acabei), cabou (acabou), celera (acelera), celerando (acelerando), certar (acertar), chei (achei), cho (acho), contece (acontece), contecer (acontecer), conteceu (aconteceu), cordava (acordava), creditei (acreditei), dianta (adianta), doro (adoro), dotada (adotada), fessora (professora), final (afinal), fundar (afundar), garrado (agarrados), garrou (agarrou), gora (agora), gual (igual), gualzim (igualzinho), guenta (aguenta), guentando (aguentando), guentar (aguentar), guento (aguento), guentou (aguentou), inda (ainda), judar (ajudar), lambique (alambique), laranjado (alaranjado), lisou (alisou), magina (imagina), mamentar (amamentar), manhã (amanhã), marelo (amarelo), marrava (amarrava), migão (amigão aumentativo), mor (amor), ném (neném), panhava (apanhava), parece (aparece), pareceu (apareceu), partamento (apartamento), pelido (apelido), perta (aperta), pertar (apertar), pertei (apertei), pesar (apesar), pinhada (apinhada), proveita (aproveita), proveitei (aproveitei), proveitou (aproveitou), proveitando (aproveitando), proveitei (aproveitei), purra (empurra), qui (daqui), rancaram (arrancaram), rancava (arrancava), rancou (arrancou), ranjar (arranjar), ranjasse (arranjasse), ranjou (arranjou), rebentando (arrebentando), rebentar (arrebentar), regaço (arregaços), rorosa (horrorosa), roz (arroz), rumaram (arrumaram), sobiando (assobiando),té (até), té (até), teirinho (inteirinho), teja (esteja), tendeu (entendeu), tendi (entendi), testino (intestino), tradinha (entradinha), trapalha (atrapalha), trapalhado (atrapalhado), trapalhou (atrapalhou), travessa (atravessa), travessadinho (atravessadinho), trevida (atrevida), trevido (atrevido), trevidão (atrevidão), vó (avó), vô (avô)"""

```
    regex_apostr = r"(\w+’(?!\n))\s(\w+(?!\n))"
    regex_excep= r"([A-Za-z]+’)([A-Za-z]+)"
    texto = texto.replace("""", "’")
    texto = re.sub(regex_apostr, r"\1\2", texto)
    formas_conv = re.sub(regex_apostr, r"\1\2", formas_conv)
    tuplas  =  re.findall(r"(\w+|\w+\s\w+|\w+’|\w+’\s?\w+|\w+\s\w+\s\w+’|\w+\s?\w+’)\s?\((\w+|\w+\s\w+)\)",
formas_conv)
    tuplas=  [(x[0].strip(), x[1]) for x in tuplas]
    dicio = dict(tuplas)
    texto = " ".join([dicio[p] if p in dicio else p for p in texto.split(' ')])
    texto= re.sub(regex_excep, r'\1 \2', texto)
    formas_conv = re.sub(regex_excep, r"\1 \2", formas_conv)
    tuplas  =  re.findall(r"(\w+|\w+\s\w+|\w+’|\w+’\s?\w+|\w+\s\w+\s\w+’|\w+\s?\w+’)\s?\((\w+|\w+\s\w+)\)",
formas_conv)
    dicio = dict(tuplas)
    texto = texto.replace('\n', '\n$ ')
    texto =  '\n'.join([dicio[x] if x in dicio else x for x in texto.split(' ')])
    texto = texto.replace('\n', ' ')
    texto = texto.split('$')
```

```
    texto = '\n'.join([x.strip() for x in texto])
    texto = texto.replace('o'', 'olha').replace('pa'', 'para')\
        .replace('Vix'', 'Vixe').replace('No'', 'Nossa').replace('pr'', 'para')\
            .replace('n'', 'não').replace('e'', 'ele').replace('Nu'', 'Nossa')
    texto = re.sub(r'i?-?_?COB_?s?r?|i?-?_?COM_?s?r?|i?-?_?APC_?s?r?|i?-?_?CMM_?s?r?|i?-
?_?TOP_?s?r?=|i?-?_?TPL_?s?r?|i?-?_?APT_?s?r?|i?-?_?PAR_?s?r?=|i?-?_?PAR_?s?r?|i?-
?_?INT_?s?r?|i?-?_?SCA_?s?r?|i?-?_?AUX_?s?r?|i?-?_?PHA_?s?r?|i?-?_?ALL_?s?r?|i?-
?_?CNT_?s?r?|i?-?_?DCT_?s?r?|i?-?_?EXP_?s?r?|i?-?_?DCT_?s?r?', '', texto)
    texto = texto.strip()
    return texto
file_1 = '\n'.join([x for x in os.listdir() if x.endswith('xml')])
utterances = []
participant = []
start_time = []
end_time = []
audio = []
for filename in file_1.split():
    with open(filename, 'r', encoding="utf-8") as content:
        tree = ET.parse(content)
        root = tree.getroot()
        for y in root.iter('UNIT'):
            utterances.append(y.text.strip())
            participant.append(y.get('speaker'))
            start_time.append((y.get('startTime')))
            end_time.append((y.get('endTime')))
            audio.append(filename)
            start_time_f = re.findall(r'\d+.\d+', '\n'.join(start_time))
            end_time_f = re.findall(r'\d+.\d+', '\n'.join(end_time))
df = pd.DataFrame()
df['utterances'] = utterances
df['participant'] = participant
df['audio'] = audio
df['COB'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?COB_?s?r?=', x)))
df['COM'] =  df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?COM_?s?r?=', x)))
df['APC'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?APC_?s?r?=', x)))
df['CMM'] =  df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?CMM_?s?r?=', x)))
df['TOP_TPL'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?TOP_?s?r?=|=i?-
?_?TPL_?s?r?=', x)))
df['APT'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?APT_?s?r?=', x)))
```

```
df['PAR_PRL'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?PAR_?s?r?=|=i?-?_?PAR_?s?r?=', x)))
df['INT'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?INT_?s?r?=', x)))
df['SCA'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?SCA_?s?r?=', x)))
df['textual_units'] = df['COM'] + df['APC'] + df['CMM'] + df['TOP_TPL'] + df['APT'] + df['PAR_PRL'] + df['INT']
df['tonal_units'] = df['utterances'].apply(lambda x: len(re.findall(r'(?<!\n)/(?!\d)', x)))
df['AUX'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?AUX_?s?r?=', x)))
df['PHA'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?PHA_?s?r?=', x)))
df['ALL'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?ALL_?s?r?=', x)))
df['CNT'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?CNT_?s?r?=', x)))
df['DCT'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?DCT_?s?r?=', x)))
df['EXP'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?EXP_?s?r?=', x)))
df['INP'] = df['utterances'].apply(lambda x: len(re.findall(r'=i?-?_?DCT_?s?r?=', x)))
df['inter_utterance'] = df['utterances'].apply(lambda x: len(re.findall(r'\+', x)))
df['start_time'] = start_time_f
df['end_time'] = end_time_f
df[['start_time', 'end_time']] = df[['start_time', 'end_time']].astype('float')
df['ut_length'] = round(df['end_time'] - df['start_time'], 3)
df['cleaned_utterances'] = df['utterances'].apply(limpa_coral)
df['number_of_words'] = df['cleaned_utterances'].apply(lambda x: len(x.split()))
df['words_per_second'] = (df['number_of_words'] / df['ut_length']).round(3)
df['corpus'] = df['audio'].apply(lambda x: 'Coral_brasil' if x.startswith('b') else 'Coral_esq')
# df['cleaned_for_overlapping'] = df_doc['utterances'].apply(limpa_coral_sobreposicao)
# df['overlapping_words'] = df['cleaned_for_overlapping'].apply(lambda x: ' '.join(re.findall(r'<.+>', str(x))))
# df['cleaned_overlapping'] = df['overlapping_words'].apply(limpa_coral)
# df['number_overlapping'] = df['cleaned_overlapping'].apply(lambda x: len(x.split()))
# df.drop(['cleaned_for_overlapping', 'cleaned_overlapping'], axis = 1, inplace = True)
df['retractings'] = df['utterances'].apply(lambda x: len(re.findall(r'/(=?\d+)', x)))
df['retr_words'] = df['utterances'].apply(lambda x: sum(map(int, re.findall(r'\d+', x))))
df['TMT'] = df['utterances'].apply(lambda x: len(re.findall(r'.?&he', x)))
df['all_patterns'] = df['utterances'].apply(lambda x: ' '.join(re.findall(r'=i?_?-?\w{3}_?r?s?=', x)))
df['filtered_patterns'] = df['all_patterns'].apply(lambda x: re.sub(r'(=SCA=|=EMP=|=UNC=|=TMT=)', '', x))
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'\s+', ' ', x)).str.strip()
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'(?<=i)_?-?\s?(?=\w+)', '-', x))
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'\s+', ' ', x)).str.strip()
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'_r(?==)|_r(?=\s)', '', x))
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'\s+', ' ', x)).str.strip()
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'_s(?==)|_s(?=\s)', '', x))
```

```
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'\s+', ' ', x)).str.strip()
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'i-|i\s(?=\w+)', '', x))
df['filtered_patterns'] = df['filtered_patterns'].apply(lambda x: re.sub(r'\s+', ' ', x)).str.strip()
df['normalized_utterances'] = df['utterances'].apply(normaliza_coral)
# df['participant'].unique()
# df.query('participant in "Mailton" or participant in "*DFL:"  or participant \
#          in "Aloysio" or participant in "Regina" or participant in "*CAR:" or \
#              participant in  "Jorge" or audio in "med"')
# a = df.query('audio in "med_008_revisado.xml"')
with open('brill_00', 'rb') as f:
    tagueador = pickle.load(f)
#depuração normalização
df['normalized_utterances'] = df['normalized_utterances'].apply(lambda x: re.sub(r"o'\s?|^o'\s?|\s?o'$",
'olha', str(x), flags = re.IGNORECASE))
df['normalized_utterances'] = df['normalized_utterances'].apply(lambda x: re.sub(r"\se'\s?|^e'\s?|\s?e'$",
'ele', str(x), flags = re.IGNORECASE))
df['normalized_utterances'] = df['normalized_utterances'].apply(lambda x: re.sub(r"\sa'\s?|^a'\s?|\s?a'$",
'olha', str(x), flags = re.IGNORECASE))
df['normalized_utterances'] = df['normalized_utterances'].apply(lambda x: re.sub(r"[A-Z]{3}-r|i-[A-Z]{3}",
'', str(x), flags = re.IGNORECASE))
print('Tagging with Brill tagger - Mac-Morpho')
df['utterances_POS'] = df['normalized_utterances'].apply(lambda x:
nltk.word_tokenize(x)).apply(lambda x: tagueador.tag(x))
#depuração POS
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='sio'',\s')\w+|(?<='sio',\s')\w+|(?<='senhora',\s')\w+",'PROPESS', str(x), flags =
re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='melancia'',\s')\w+",'N', str(x),
flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"\(\'Nossa',\s\'\w+\'\),\s\(\'Senhora\',\s\'\w+\'\),", "('Nossa Senhora', 'IN'),", x))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='na',\s')\w+|(?<='no',\s')\w+",
'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='nessa',\s')\w+|(?<='nesse',\s')\w+|(?<='nisso',\s')\w+|(?<='nesses',\s')\w+|(?<='nessas',\s')
\w+", 'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='ao',\s')\w+|(?<='aos',\s')\w+",
'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='o',\s')\w+|(?<='os',\s')\w+",
'ART', str(x), flags = re.IGNORECASE))
```

```
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='da',\s')\w+|(?<='do',\s')\w+|(?<='dos',\s')\w+|(?<='das',\s')\w+|(?<='duma',\s')\w+|(?<='dum',\s')\w+|(?<='duns',\s')\w+|(?<='dumas',\s')\w+", 'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='de',\s')\w+", 'PREP', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='pelos',\s')\w+|(?<='pelo',\s')\w+|(?<='pela',\s')\w+|(?<='pelas',\s')\w+", 'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='dele',\s')\w+|(?<='deles',\s')\w+|(?<='dela',\s)\w+|(?<='delas',\s)\w+", 'PROADJ', str(x) , flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='num',\s')\w+", 'PREP|+', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='um',\s')\w+|(?<='uns',\s')\w+|(?<='uma',\s')\w+|(?<='umas',\s')\w+", 'ART', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='comigo',\s')\w+", 'PROPESS', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='por',\s')\w+", 'PREP', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='contigo',\s')\w+", 'PROPESS', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='ahn',\s')\w+|(?<='ham',\s')\w+|(?<='hum',\s')\w+|(?<='uhn',\s')\w+", 'IN', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='ah',\s')\w+|(?<='eh',\s')\w+|(?<='ih',\s')\w+|(?<='oh',\s')\w+|(?<='ô',\s')\w+|(?<='uai',\s')\w+|(?<='ué',\s')\w+", 'IN', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='nu',\s')\w+|(?<='pá',\s')\w+|(?<='parará',\s')\w+|(?<='tanãnãnã',\s')\w+|(?<='tchan',\s')\w+|(?<='tum',\s')\w+", 'IN', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='deixar',\s')\w+|(?<='deixa',\s')\w+|(?<='deixou',\s')\w+|(?<='deixei',\s')\w+|(?<='deixaram',\s')\w+|(?<='deixamos',\s')\w+|(?<='deixaria',\s')\w+|(?<='deixariam',\s')\w+|(?<='deixam',\s')\w+|(?<='deixo',\s')\w+|(?<='deixasse',\s')\w+|(?<='deixassem',\s')\w+|(?<='deixarmos',\s')\w+", 'VAUX', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='começar',\s')\w+|(?<='começa',\s')\w+|(?<='começou',\s')\w+|(?<='comecei',\s')\w+|(?<='começaram',\s')\w+|(?<='começam',\s')\w+|(?<='começaria',\s')\w+|(?<='começariam',\s')\w+|(?<='come
```

```
çamos',\s')\w+|(?<='começo',\s')\w+|(?<='começasse',\s')\w+|(?<='começassem',\s')\w+|(?<='começar
mos',\s')\w+|(?<='comece',\s')\w+|(?<='comecemos',\s')\w+", 'V', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='chegar',\s')\w+|(?<='chega',\s')\w+|(?<='chegou',\s')\w+|(?<='cheguei',\s')\w+|(?<='chegar
am',\s')\w+|(?<='chegam',\s')\w+|(?<='chegaria',\s')\w+|(?<='chegariam',\s')\w+|(?<='chegamos',\s')\w+|
(?<='chego',\s')\w+|(?<='chegasse',\s')\w+|(?<='chegassem',\s')\w+|(?<='chegarmos',\s')\w+|(?<='cheg
ue',\s')\w+|(?<='cheguemos',\s')\w+", 'V', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='ser',\s')\w+|(?<='é',\s')\w+|(?<='foi',\s')\w+|(?<='fui',\s')\w+|(?<='foram',\s')\w+|(?<='somos'
,\s')\w+|(?<='seria',\s')\w+|(?<='seriam',\s')\w+|(?<='são',\s')\w+|(?<='sou',\s')\w+|(?<='era',\s')\w+|(?<='
eram',\s')\w+|(?<='for',\s')\w+|(?<='formos',\s')\w+|(?<='fosse',\s')\w+|(?<='fóssemos',\s')\w+|(?<='sermo
s',\s')\w+", 'VAUX', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='poder',\s')\w+|(?<='pode',\s')\w+|(?<='pôde',\s')\w+|(?<='pude',\s')\w+|(?<='puderam',\s')\
w+|(?<='podemos',\s')\w+|(?<='poderia',\s')\w+|(?<='poderiam',\s')\w+|(?<='podem',\s')\w+|(?<='posso',\
s')\w+|(?<='podia',\s')\w+|(?<='podiam',\s')\w+|(?<='pudesse',\s')\w+|(?<='pudessem',\s')\w+|(?<='poder
mos',\s')\w+", 'VAUX', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='estar',\s')\w+|(?<='está',\s')\w+|(?<='esteve',\s')\w+|(?<='estive',\s')\w+|(?<='estiveram',\s')
\w+|(?<='estamos',\s')\w+|(?<='estaria',\s')\w+|(?<='estariam',\s')\w+|(?<='estão',\s')\w+|(?<='estou',\s')\
w+|(?<='estava',\s')\w+|(?<='estivemos',\s')\w+|(?<='estivesse',\s')\w+|(?<='estivéssemos',\s')\w+|(?<='
estivessem',\s')\w+|(?<='estarmos',\s')\w+", 'VAUX', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='ir',\s')\w+|(?<='vai',\s')\w+|(?<='foi',\s')\w+|(?<='fui',\s')\w+|(?<='foram',\s')\w+|(?<='vamos',
\s')\w+|(?<='iria',\s')\w+|(?<='iriam',\s')\w+|(?<='vão',\s')\w+|(?<='vou',\s')\w+|(?<='ía',\s')\w+|(?<='fomos'
,\s')\w+|(?<='fosse',\s')\w+|(?<='fóssemos',\s')\w+|(?<='iria',\s')\w+|(?<='vamos',\s')\w+", 'VAUX', str(x),
flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='nũ',\s')\w+|(?<='né',\s')\w+|(?<='aí',\s')\w+", 'ADV', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='Whatsapp',\s')\w+|(?<='Instagram',\s')\w+|(?<='Facebook',\s')\w+|(?<='big',\s')\w+|(?<='br
other',\s')\w+|(?<='buffet',\s')\w+|(?<='feedback',\s')\w+|(?<='fair',\s')\w+|(?<='play',\s')\w+", '|EST', str(x),
flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='open',\s')\w+|(?<='over',\s')\w+|(?<='photoshop',\s')\w+|(?<='pop',\s')\w+|(?<='plus',\s')\w+
|(?<='réveillon',\s')\w+|(?<='sexy',\s')\w+|(?<='serial',\s')\w+|(?<='killer',\s')\w+", '|EST', str(x), flags =
re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='shopping',\s')\w+|(?<='short',\s')\w+|(?<='show',\s')\w+|(?<='smartphone',\s')\w+|(?<='soft
```

```
ware',\s')\w+|(?<='telemarketing',\s')\w+|(?<='videogame',\s')\w+|(?<='tablet',\s')\w+|(?<='Windows',\s')\
w+", '|EST', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='yes',\s')\w+|(?<='vip',\s')\w+|(?<='web',\s')\w+|(?<='smartphone',\s')\w+|(?<='slide',\s')\w+|
(?<='states',\s')\w+|(?<='videogame',\s')\w+|(?<='online',\s')\w+|(?<='office',\s')\w+|(?<='offline',\s')\w+",
'|EST', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x:
re.sub(r"(?<='ficar',\s')\w+|(?<='fica',\s')\w+|(?<='ficou',\s')\w+|(?<='fiquei',\s')\w+|(?<='ficaram',\s')\w+|(?
<='ficamos',\s')\w+|(?<='ficaria',\s')\w+|(?<='ficariam',\s')\w+|(?<='ficam',\s')\w+|(?<='fico',\s')\w+|(?<='fic
asse',\s')\w+|(?<='ficassem',\s')\w+|(?<='ficarmos',\s')\w+", 'VAUX', str(x), flags = re.IGNORECASE))
df['utterances_POS'] = df['utterances_POS'].apply(lambda x: re.sub(r"(?<='né',\s')\w+", 'ADV', str(x),
flags = re.IGNORECASE))
#soma etiquetas dialógicas em uma única coluna - mas deixa as que já existem lá
df_correc_cesq = df.query('corpus in "Coral_esq"')
df_correc_cesq_1 = df_correc_cesq.query('audio in "med_007.xml" or audio in "med_019.xml" or audio
in "med_020.xml"')
df_correc_cesq_1['AUX'] = df_correc_cesq_1['PHA'] + df_correc_cesq_1['ALL'] +
df_correc_cesq_1['CNT'] + df_correc_cesq_1['DCT'] + df_correc_cesq_1['EXP'] +
df_correc_cesq_1['EXP'] + df_correc_cesq_1['INP']
df_correc_cesq_2 = df_correc_cesq.query('audio in "med_008.xml" or audio in "med_013.xml" or audio
in "med_015.xml"')
df_correc_cesq = pd.concat([df_correc_cesq_1, df_correc_cesq_2])
df_correc_cb = df.query('corpus in "Coral_brasil"')
df_correc_cb['AUX'] = df_correc_cb['PHA'] + df_correc_cb['ALL'] + df_correc_cb['CNT'] +
df_correc_cb['DCT'] + df_correc_cb['EXP'] + df_correc_cb['EXP'] + df_correc_cb['INP']
df = pd.concat([df_correc_cb, df_correc_cesq], ignore_index= True)
#seleciona apenas os pacientes do C-ORAL-ESQ (utilizados com COSTA, 2022)
df = df.query('participant in "MIR" or participant in "CLA" or participant in "GLE" \
      or participant in "DAN" or participant in "MAA" or participant in "VIT" or \
     participant in "Mailton" or participant in "*DFL:" or participant in "Aloysio" \
     or participant in "Regina" or participant in "*CAR:" or participant in "Jorge"')
df['phrases_before_retractings'] = df['utterances'].apply(lambda x: '
'.join(re.findall(r".\w+\s.\w+\s(?=\[/\d\])", x)))
df['phrases_after_retractings'] = df['utterances'].apply(lambda x: '
'.join(re.findall(r"(?<=\[/\d\]\s)\w+\s\w+", x)))
df['dist_retractings'] = df['utterances'].apply(lambda x: ' '.join(re.findall(r'.\w+(?=\s\[\/\d\])|.\w+(?=\s/\d)',
x)))
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
   df_cb = df.query('corpus in "Coral_brasil"')
   dist_ret_cb = df_cb['dist_retractings'].tolist()
```

```
    list_participants_cb = df_cb['participant'].to_list()
    list_ret_cb= []
    for x in dist_ret_cb:
        for y in x.split():
            list_ret_cb.append(y)
    list_ret_cb = '\n'.join(list_ret_cb)
    list_ret_cb = re.sub('\[|\]|\"|,', '', list_ret_cb)
    list_ret_cb = [x.lower() for x in list_ret_cb.splitlines() if len(x) > 0]
    list_ret_df_cb = pd.DataFrame([x.strip() for x in list_ret_cb], columns=['retractings_full'])
    list_ret_df_cb['corpus'] = 'Coral_brasil'
    #cesq
    df_cesq = df.query('corpus in "Coral_esq"')
    dist_ret_cesq = df_cesq['dist_retractings'].tolist()
    list_participants_cesq = df_cesq['participant'].to_list()
    list_ret_cesq= []
    for x in dist_ret_cesq:
        for y in x.split():
            list_ret_cesq.append(y)
    list_ret_cesq = '\n'.join(list_ret_cesq)
    list_ret_cesq = re.sub('\[|\]|\"|,', '', list_ret_cesq)
    list_ret_cesq = [x.lower() for x in list_ret_cesq.splitlines() if len(x) > 0]
    list_ret_df_cesq = pd.DataFrame([x.strip() for x in list_ret_cesq], columns=['retractings_full'])
    list_ret_df_cesq['corpus'] = 'Coral_esq'
    list_ret_df = pd.concat([list_ret_df_cb, list_ret_df_cesq])
else:
    dist_ret = df['dist_retractings'].tolist()
    list_participants = df['participant'].to_list()
    list_ret= []
    for x in dist_ret:
        for y in x.split():
            list_ret.append(y)
    list_ret = '\n'.join(list_ret)
    list_ret = re.sub('\[|\]|\"|,', '', list_ret)
    list_ret = [x.lower() for x in list_ret.splitlines() if len(x) > 0]
    list_ret_df = pd.DataFrame([x.strip() for x in list_ret], columns=['retractings_full'])
list_ret_df['retractings'] = list_ret_df['retractings_full'].apply(lambda x: re.sub(r"&|-|<|>|\=", '', x))
list_ret_df['retractings_syl'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?a
u.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?",
str(x), flags = re.IGNORECASE)))
```

```
list_ret_df['oclus_des'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r"^p|^k|^k$|^c$|^ca|^co|^cu|^cã|^cô|^câ|^cõ|^cú|^q.?.?|^cr|^te(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|
b|d|g|v|z|j|m|n|u|ú|ó|o|ô|õ)|^té|^tê|^ta|^tá|^tã|^tâ|^to|^tô|^tu|^tú|^tr", x)))
list_ret_df['oclus_voz'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r"^b|^ga|^go|^gu|^gão|^gá|^gó|^gú|^gâ|^gr|^g$|^da|^dá|^dã|^dõ|^dô|^dú|^du|^do|^dr|^de
(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)", x)))
list_ret_df['laterais'] = list_ret_df['retractings'].apply(lambda x: ' '.join(re.findall(r"(^l|^lh)", x)))
list_ret_df['fricativas_des'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r"(^s|^x|^r|^ch|^f|^ce|^ci|^cê|^cí|^cé)", x)))
list_ret_df['fricativas_voz'] = list_ret_df['retractings'].apply(lambda x: ' '.join(re.findall(r'(^z|^j|^v|^ge|^gi)',
x)))
list_ret_df['oclus_nas'] = list_ret_df['retractings'].apply(lambda x: ' '.join(re.findall(r'(^m|^n)', x)))
list_ret_df['africadas_des'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r'^te(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|a|á|ã|à|â|e|é|ê|i|í|o|ó|õ|ô|u|ú)|^ti|^t$', x)))
list_ret_df['africadas_voz'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r'^de(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)|^di|^d$', x)))
list_ret_df['vogais'] = list_ret_df['retractings'].apply(lambda x: '
'.join(re.findall(r"^a|^e|^i|^o|^u|^ã|^õ|^à|^á|^â|^ô|^ó|^é|^ê|^ú|^h", x)))
list_ret_df = list_ret_df.loc[list_ret_df['retractings_full'] != 'xxx']
list_ret_df = list_ret_df.loc[list_ret_df['retractings_full'] != 'xxxx']
list_ret_df = list_ret_df.loc[list_ret_df['retractings_full'] != 'yyy']
list_ret_df = list_ret_df.loc[list_ret_df['retractings_full'] != 'yyyy']
list_ret_df = list_ret_df.loc[list_ret_df['retractings_full'] != 'hhh']
list_ret_df['retractings'] = list_ret_df['retractings'].apply(lambda x: x[:3] if len(x) > 3 else x)
list_ret_df['oclus_des'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r"^p|^k|^k$|^c$|^ca|^co|^cu|^cã|^cô|^câ|^cõ|^cú|^q.?.?|^cr|^te(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|
d|g|v|z|j|m|n|u|ú|ó|o|ô|õ)|^té|^tê|^ta|^tá|^tã|^tâ|^to|^tô|^tu|^tú|^tr", str(x))))
list_ret_df['oclus_voz'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r"^b|^ga|^go|^gu|^gão|^gá|^gó|^gú|^gâ|^gr|^g$|^da|^dá|^dã|^dõ|^dô|^dú|^du|^do|^dr|^de(?
=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)", x)))
list_ret_df['laterais'] = list_ret_df['retractings'].apply(lambda x: len(re.findall(r"(^l|^lh)", x)))
list_ret_df['fricativas_des'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r"(^s|^x|^r|^ch|^f|^ce|^ci|^cê|^cí|^cé)", x)))
list_ret_df['fricativas_voz'] = list_ret_df['retractings'].apply(lambda x: len(re.findall(r'(^z|^j|^v|^ge|^gi)',
x)))
list_ret_df['oclus_nas'] = list_ret_df['retractings'].apply(lambda x: len(re.findall(r'(^m|^n)', x)))
list_ret_df['africadas_des'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r'^te(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|a|á|ã|à|â|e|é|ê|i|í|o|ó|õ|ô|u|ú)|^ti|^t$', x)))
list_ret_df['africadas_voz'] = list_ret_df['retractings'].apply(lambda x:
len(re.findall(r'^de(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)|^di|^d$', x)))
```

```
list_ret_df['vogais'] = list_ret_df['retractings'].apply(lambda x: len(re.findall(r"^a|^e|^i|^o|^u|^ã|^õ|^à|^á|^â|^ô|^ó|^é|^ê|^ú|^h", x)))
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
    retracted_words = pd.DataFrame(list_ret_df.groupby('corpus')['retractings_full'].value_counts())
    retracted_words.columns = ['Frequência']
    retracted_words.reset_index(inplace=True)
    retracted_words.columns = ['corpus', 'retractings', 'Frequência']
else:
    contagem_retratadas = pd.DataFrame(list_ret_df['retractings_full'].value_counts())
    contagem_retratadas.reset_index(inplace=True)
    contagem_retratadas.columns = ['retractings', 'Frequência']
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
    plt.figure(figsize =(9, 5), dpi = 200)
    a = sns.barplot(data = retracted_words.sort_values(by = 'Frequência', ascending =False)[:30], x = 'retractings', y = 'Frequência', hue = 'corpus')
    a.set_title('Palavras retratadas nos corpora', fontsize = 18)
    a.set_ylabel('Quantidade', fontsize = 15)
    a.set_xlabel('Classe do segmento', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13)
else:
    plt.figure(figsize =(9, 5), dpi = 200)
    a = sns.barplot(data = contagem_retratadas[:30], x = 'retractings', y = 'Frequência')
    a.set_title('Frequência de palavras retratadas', fontsize = 18)
    a.set_ylabel('Quantidade', fontsize = 15)
    a.set_xlabel('Classe do segmento', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13)
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
    list_ret_df_visu_cb = list_ret_df.query('corpus in "Coral_brasil"')
    list_ret_df_visu_cb.drop(list_ret_df_visu_cb.loc[:, 'retractings_full': 'retractings_syl'], axis = 1, inplace=True)
    # list_ret_df_visu = list_ret_df_visu_cb.loc[:, 'oclus_des':]
    # list_ret_df_visu_cb.drop('corpus', axis =1, inplace= True)
    list_ret_df_visu_cb = list_ret_df_visu_cb.melt()
    list_ret_df_visu_cb = pd.DataFrame(list_ret_df_visu_cb.groupby('variable')['value'].sum())
    list_ret_df_visu_cb.reset_index(inplace=True)
    list_ret_df_visu_cb['corpus'] = 'Coral_brasil'
```

```
    #cesq
    list_ret_df_visu_cesq = list_ret_df.query('corpus in "Coral_esq"')
    list_ret_df_visu_cesq.drop(list_ret_df_visu_cesq.loc[:, 'retractings_full': 'retractings_syl'], axis = 1,
inplace=True)
    # list_ret_df_visu = list_ret_df_visu_cesq.loc[:, 'oclus_des':]
    # list_ret_df_visu_cesq.drop('corpus', axis =1, inplace= True)
    list_ret_df_visu_cesq = list_ret_df_visu_cesq.melt()
    list_ret_df_visu_cesq = pd.DataFrame(list_ret_df_visu_cesq.groupby('variable')['value'].sum())
    list_ret_df_visu_cesq.reset_index(inplace=True)
    list_ret_df_visu_cesq['corpus'] = 'Coral_esq'
    list_ret_df_classe = pd.concat([list_ret_df_visu_cb, list_ret_df_visu_cesq], ignore_index=True)
    list_ret_df_classe.sort_values(by ='value', ascending = False, inplace=True)
    list_ret_df_classe.columns = ['Classe_do_segmento', 'Frequência', 'corpus']
    sns.set_style('whitegrid')
    plt.figure(figsize =(7, 5), dpi = 200)
    a = sns.barplot(data = list_ret_df_classe,x = 'Classe_do_segmento', y = 'Frequência', hue = 'corpus',
estimator = sum )
    a.set_title('Classe do segmento inicial do retracting por corpora', fontsize = 18)
    a.set_ylabel('Quantidade', fontsize = 15)
    a.set_xlabel('Classe do segmento', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13)
else:
    list_ret_df_visu = list_ret_df.copy()
    list_ret_df_visu.drop(list_ret_df_visu.loc[:, 'retractings_full': 'retractings_syl'], axis = 1, inplace=True)
    # list_ret_df_visu = list_ret_df_visu.loc[:, 'oclus_des':]
    # list_ret_df_visu.drop('corpus', axis =1, inplace= True)
    list_ret_df_visu = list_ret_df_visu.melt()
    list_ret_df_visu = pd.DataFrame(list_ret_df_visu.groupby('variable')['value'].sum())
    list_ret_df_visu.reset_index(inplace=True)
    # list_ret_df_visu['corpus'] = 'Coral_brasil'
    list_ret_df_visu.columns = ['Classe_do_segmento', 'Frequência']
    sns.set_style('whitegrid')
    plt.figure(figsize =(7, 5), dpi = 200)
    a = sns.barplot(data = list_ret_df_visu.sort_values(by = 'Frequência', ascending = False),x =
'Classe_do_segmento', y = 'Frequência', palette = 'inferno')
    a.set_title('Classe do segmento inicial do retracting por corpora', fontsize = 18)
    a.set_ylabel('Quantidade', fontsize = 15)
    a.set_xlabel('Classe do segmento', fontsize = 15)
```

```
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13)
#medidas
df_speech_m = round(df.groupby(['corpus', 'participant']).agg({'ut_length': ['sum', 'mean', 'std', 'min', 'max'], 'number_of_words':['sum', 'mean', 'std', 'min', 'max'], 'words_per_second': ['mean', 'min', 'max', 'std']}), 2)
try:
    df_inform_m = pd.DataFrame(df.groupby(['corpus','participant']).agg({'COB': ['sum', 'mean', 'max', 'min'], 'COM': ['sum', 'mean', 'max', 'min'], \
                            'APC': ['sum', 'mean', 'max', 'min'], 'CMM': ['sum', 'mean', 'max', 'min'], \
                            'CMM': ['sum', 'mean', 'max', 'min'], 'TOP_TPL': ['sum', 'mean', 'max', 'min'],\
                            'APT': ['sum', 'mean', 'max', 'min'], 'PAR_PRL':['sum', 'mean', 'max', 'min'], \
                            'INT': ['sum', 'mean', 'max', 'min'], 'SCA': ['sum', 'mean', 'max', 'min'], \
                            'textual_units': ['sum', 'mean', 'max', 'min'], 'tonal_units': ['sum', 'mean', 'max', 'min'], \
                            'AUX': ['sum', 'mean', 'max', 'min'], 'PHA': ['sum', 'mean', 'max', 'min'], \
                            'ALL': ['sum', 'mean', 'max', 'min'], 'CNT': ['sum', 'mean', 'max', 'min'], \
                            'DCT': ['sum', 'mean', 'max', 'min'], \
                            'EXP': ['sum', 'mean', 'max', 'min']}))
except:
    pass
df_inform_m = df_inform_m.round(2)
grouped_filt_patterns = df.groupby(['corpus','participant'])['filtered_patterns'].value_counts()
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
    if df.textual_units.sum() > 0:
        counted_patterns = pd.DataFrame(df.groupby('corpus')['filtered_patterns'].value_counts())
        counted_patterns.columns = ['Frequência']
        counted_patterns.reset_index(inplace=True)
        counted_patterns.columns = ['corpus', 'Padrões_inf', 'Frequência']
        counted_patterns['Padrões_inf'] = counted_patterns['Padrões_inf'].str.replace('=', '').str.strip()
        counted_patterns['Padrões_inf']= counted_patterns['Padrões_inf'].apply(lambda x: '0' if len(x) < 3 else x)
        counted_patterns = counted_patterns.query('Padrões_inf != "0"')
        sns.set_style('whitegrid')
        plt.figure(dpi = 200, figsize = (9, 5))
        plt.xticks(rotation=90)
        b = sns.barplot(data = counted_patterns.sort_values(by = 'Frequência', ascending= False)[:15], x = 'Padrões_inf', y = "Frequência", hue = 'corpus')
        b.set_title('Padrões informacionais mais frequentes', fontsize = 16)
```

```
        b.set_ylabel("Frequência", fontsize = 15)
        b.set_xlabel('Padrões informacionais',fontsize=14)
        b.tick_params(labelsize=15)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
else:
    try:
        counted_patterns = pd.DataFrame(df['filtered_patterns'].value_counts())
        counted_patterns.columns = ['Frequência']
        counted_patterns.reset_index(inplace=True)
        counted_patterns.columns = ['Padrões_inf', 'Frequência']
        counted_patterns = counted_patterns.query('Padrões_inf != ""')
        sns.set_style('whitegrid')
        plt.figure(dpi = 200, figsize = (6, 4))
        plt.xticks(rotation=90)
        b = sns.barplot(data = counted_patterns[:10], x = 'Padrões_inf', y = 'Frequência')
        b.set_title('Padrões informacionais mais frequentes', fontsize = 16)
        b.set_ylabel("Frequência", fontsize = 15)
        b.set_xlabel('Padrões informacionais',fontsize=14)
        b.tick_params(labelsize=15)
    except:
        pass
print('By: José Carlos Costa \ Let me know if I can help you with something! \
      \n whatsapp: +55 31 98924 1307 \n \
      email: carlosjuniorcosta1@gmail.com')
try:
    df1 = df.query('COB > 0')
    df_cb_bal = df1.query('corpus in "Coral_brasil"')
    df_cesq_bal = df1.query('corpus in "Coral_esq"')
except:
    pass
try:
    df_cb_cob1 = df_cb_bal.query('COB == 1')
    df_cb_cob2 = df_cb_bal.query('COB == 2')
    df_cb_cob3 = df_cb_bal.query('COB == 3')
    df_cb_cob4 = df_cb_bal.query('COB == 4')
    df_cb_cob5= df_cb_bal.query('COB == 5')
    df_cb_cob6 = df_cb_bal.query('COB == 6')
    df_cb_cob7 = df_cb_bal.query('COB == 7')
    df_cb_cob8 = df_cb_bal.query('COB == 8')
    df_cb_cob9 = df_cb_bal.query('COB == 9')
```

```
    df_cb_cob10 = df_cb_bal.query('COB == 10')
    df_cb_cob11 = df_cb_bal.query('COB == 11')
    df_cb_cob12 = df_cb_bal.query('COB == 12')
    df_cb_cob13 = df_cb_bal.query('COB == 13')
    df_cb_cob14 = df_cb_bal.query('COB == 14')
    df_cb_cob15 = df_cb_bal.query('COB == 15')
except:
    pass
try:
    df_cb_cob1 = df_cb_bal.query('COB == 1')
    df_cb_cob2 = df_cb_bal.query('COB == 2')
    df_cb_cob3 = df_cb_bal.query('COB == 3')
    df_cb_cob4 = df_cb_bal.query('COB == 4')
    df_cb_cob5= df_cb_bal.query('COB == 5')
    df_cb_cob6 = df_cb_bal.query('COB == 6')
    df_cb_cob7 = df_cb_bal.query('COB == 7')
    df_cb_cob8 = df_cb_bal.query('COB == 8')
    df_cb_cob9 = df_cb_bal.query('COB == 9')
    df_cb_cob10 = df_cb_bal.query('COB == 10')
    df_cb_cob11 = df_cb_bal.query('COB == 11')
    df_cb_cob12 = df_cb_bal.query('COB == 12')
    df_cb_cob13 = df_cb_bal.query('COB == 13')
    df_cb_cob14 = df_cb_bal.query('COB == 14')
    df_cb_cob15 = df_cb_bal.query('COB == 15')
except:
    pass
try:
    df_cesq_cob1 = df_cesq_bal.query('COB == 1')
    df_cesq_cob2 = df_cesq_bal.query('COB == 2')
    df_cesq_cob3 = df_cesq_bal.query('COB == 3')
    df_cesq_cob4 = df_cesq_bal.query('COB == 4')
    df_cesq_cob5= df_cesq_bal.query('COB == 5')
    df_cesq_cob6 = df_cesq_bal.query('COB == 6')
    df_cesq_cob7 = df_cesq_bal.query('COB == 7')
    df_cesq_cob8 = df_cesq_bal.query('COB == 8')
    df_cesq_cob9 = df_cesq_bal.query('COB == 9')
    df_cesq_cob10 = df_cesq_bal.query('COB == 10')
    df_cesq_cob11 = df_cesq_bal.query('COB == 11')
    df_cesq_cob12 = df_cesq_bal.query('COB == 12')
    df_cesq_cob13 = df_cesq_bal.query('COB == 13')
```

```
    df_cesq_cob14 = df_cesq_bal.query('COB == 14')
    df_cesq_cob15 = df_cesq_bal.query('COB == 15')
except:
    pass
if len(df_cb_cob1) and len(df_cesq_cob1) > 0:
   df_cob1_bal = pd.concat([df_cb_cob1, df_cesq_cob1])
   valor_cob1 = min(df_cob1_bal.corpus.value_counts())
   df_cob1_bal = df_cob1_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob1)))
   if len(df_cob1_bal) > 0:
      df_bal = df_cob1_bal.copy()
if len(df_cb_cob2) and len(df_cesq_cob2) > 0:
   df_cob2_bal = pd.concat([df_cb_cob2, df_cesq_cob2])
   valor_cob2 = min(df_cob2_bal.corpus.value_counts())
   df_cob2_bal = df_cob2_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob2)))
   if len(df_cob2_bal) > 0:
      df_bal = pd.concat([df_cob1_bal, df_cob2_bal])
if len(df_cb_cob3) and len(df_cesq_cob3) > 0:
   df_cob3_bal = pd.concat([df_cb_cob3, df_cesq_cob3])
   valor_cob3 = min(df_cob3_bal.corpus.value_counts())
   df_cob3_bal = df_cob3_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob3)))
   if len(df_cob3_bal) > 0:
      df_bal = pd.concat([df_bal, df_cob3_bal])
if len(df_cb_cob4) and len(df_cesq_cob4) > 0:
   df_cob4_bal = pd.concat([df_cb_cob4, df_cesq_cob4])
   valor_cob4 = min(df_cob4_bal.corpus.value_counts())
   df_cob4_bal = df_cob4_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob4)))
   if len(df_cob4_bal) > 0:
      df_bal = pd.concat([df_bal, df_cob4_bal])
if len(df_cb_cob5) and len(df_cesq_cob5) > 0:
   df_cob5_bal = pd.concat([df_cb_cob5, df_cesq_cob5])
   valor_cob5 = min(df_cob5_bal.corpus.value_counts())
   df_cob5_bal = df_cob5_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob5)))
   if len(df_cob5_bal) > 0:
      df_bal = pd.concat([df_bal, df_cob5_bal])
if len(df_cb_cob6) and len(df_cesq_cob6) > 0:
```

```
  df_cob6_bal = pd.concat([df_cb_cob6, df_cesq_cob6])
  valor_cob6 = min(df_cob6_bal.corpus.value_counts())
  df_cob6_bal = df_cob6_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob6)))
  if len(df_cob6_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob6_bal])
if len(df_cb_cob7) and len(df_cesq_cob7) > 0:
  df_cob7_bal = pd.concat([df_cb_cob7, df_cesq_cob7])
  valor_cob7 = min(df_cob7_bal.corpus.value_counts())
  df_cob7_bal = df_cob7_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob7)))
  if len(df_cob7_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob7_bal])
if len(df_cb_cob8) and len(df_cesq_cob8) > 0:
  df_cob8_bal = pd.concat([df_cb_cob8, df_cesq_cob8])
  valor_cob8 = min(df_cob8_bal.corpus.value_counts())
  df_cob8_bal = df_cob8_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob8)))
  if len(df_cob8_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob8_bal])
if len(df_cb_cob9) and len(df_cesq_cob9) > 0:
  df_cob9_bal = pd.concat([df_cb_cob9, df_cesq_cob9])
  valor_cob9 = min(df_cob9_bal.corpus.value_counts())
  df_cob9_bal = df_cob9_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob9)))
  if len(df_cob9_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob9_bal])
if len(df_cb_cob10) and len(df_cesq_cob10) > 0:
  df_cob10_bal = pd.concat([df_cb_cob10, df_cesq_cob10])
  valor_cob10 = min(df_cob10_bal.corpus.value_counts())
  df_cob10_bal = df_cob10_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob10)))
  if len(df_cob10_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob10_bal])
if len(df_cb_cob11) and len(df_cesq_cob11) > 0:
  df_cob11_bal = pd.concat([df_cb_cob11, df_cesq_cob11])
  valor_cob11 = min(df_cob11_bal.corpus.value_counts())
  df_cob11_bal = df_cob11_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob11)))
  if len(df_cob11_bal) > 0:
```

```
    df_bal = pd.concat([df_bal, df_cob11_bal])
if len(df_cb_cob12) and len(df_cesq_cob12) > 0:
  df_cob12_bal = pd.concat([df_cb_cob12, df_cesq_cob12])
  valor_cob12 = min(df_cob12_bal.corpus.value_counts())
  df_cob12_bal = df_cob12_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob12)))
  if len(df_cob12_bal) > 0:
    df_bal = pd.concat([df_bal, df_bal12_bal])
if len(df_cb_cob13) and len(df_cesq_cob13) > 0:
  df_cob13_bal = pd.concat([df_cb_cob13, df_cesq_cob13])
  valor_cob13 = min(df_cob13_bal.corpus.value_counts())
  df_cob13_bal = df_cob13_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob13)))
  if len(df_cob13_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob13_bal])
if len(df_cb_cob14) and len(df_cesq_cob14) > 0:
  df_cob14_bal = pd.concat([df_cb_cob14, df_cesq_cob14])
  valor_cob14 = min(df_cob14_bal.corpus.value_counts())
  df_cob14_bal = df_cob14_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob14)))
  if len(df_cob14_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob14_bal])
if len(df_cb_cob15) and len(df_cesq_cob15) > 0:
  df_cob15_bal = pd.concat([df_cb_cob15, df_cesq_cob15])
  valor_cob15 = min(df_cob15_bal.corpus.value_counts())
  df_cob15_bal = df_cob15_bal.groupby('corpus', group_keys = False).apply(lambda x:
x.sample(min(len(x), valor_cob15)))
  if len(df_cob15_bal) > 0:
    df_bal = pd.concat([df_bal, df_cob15_bal])
df_bal = df_bal.query('COB < 6')
if len(df_bal) > 0:
   plt.figure(figsize =(8, 5), dpi = 200)
   a = sns.barplot(data = df_bal, x = 'COB', y = 'COM', hue = 'corpus', ci = False)
   a.set_title('Comentários por amostra', fontsize = 18)
   a.set_ylabel('Média de Comentários', fontsize = 15)
   a.set_xlabel('COBs', fontsize = 15)
   a.tick_params(labelsize = 15)
   plt.xticks(rotation = 90)
   plt.legend(frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
   plt.figure(figsize =(8, 5), dpi = 200)
```

```
a = sns.barplot(data = df_bal, x = 'COB', y = 'APC', hue = 'corpus', ci = False)
a.set_title('Apêndices de Comentário por COB', fontsize = 18)
a.set_ylabel('Média de Apêndices de Comentários', fontsize = 15)
a.set_xlabel('COBs', fontsize = 15)
a.tick_params(labelsize = 15)
plt.xticks(rotation = 90)
plt.legend(frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
sns.set_style('whitegrid')
plt.figure(figsize =(8, 5), dpi = 200)
a = sns.barplot(data = df_bal, x = 'COB', y = 'TOP_TPL', hue = 'corpus', ci = False)
a.set_title('Tópicos por COB', fontsize = 18)
a.set_ylabel('Média de Tópicos', fontsize = 15)
a.set_xlabel('COBs', fontsize = 15)
a.tick_params(labelsize = 15)
plt.xticks(rotation = 90)
plt.legend(frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
plt.figure(figsize =(8, 5), dpi = 200)
a = sns.barplot(data = df_bal, x = 'COB', y = 'APT', hue = 'corpus', ci = False)
a.set_title('Apêndices de Tópico por COB', fontsize = 18)
a.set_ylabel('Média de Apêndice de Tópicos', fontsize = 15)
a.set_xlabel('COBs', fontsize = 15)
a.tick_params(labelsize = 15)
plt.xticks(rotation = 90)
plt.legend(frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
plt.figure(figsize =(8, 5), dpi = 200)
a = sns.barplot(data = df_bal, x = 'COB', y = 'PAR_PRL', hue = 'corpus', ci = False)
a.set_title('Parentéticos por COB', fontsize = 18)
a.set_ylabel('Média de Parentéticos', fontsize = 15)
a.set_xlabel('COBs', fontsize = 15)
a.tick_params(labelsize = 15)
plt.xticks(rotation = 90)
plt.legend(frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
plt.figure(figsize =(8, 5), dpi = 200)
a = sns.barplot(data = df_bal, x = 'COB', y = 'INT', hue = 'corpus', ci = False)
a.set_title('Introdutores Locutivos por COB', fontsize = 18)
a.set_ylabel('Média de Introdutores Locutivos', fontsize = 15)
a.set_xlabel('COBs', fontsize = 15)
a.tick_params(labelsize = 15)
plt.xticks(rotation = 90)
plt.legend(loc="upper right", frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
```

```
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'inter_utterance', hue = 'corpus', ci = False, estimator =
np.mean)
    a.set_title('Enunciados interrompidos por COB', fontsize = 18)
    a.set_ylabel('Média de enunciados interrompidos', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper left", frameon=True, fontsize=13, bbox_to_anchor=(0.5, 0.5, 0.78, 0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'words_per_second', hue = 'corpus', ci = False, estimator
= np.mean)
    a.set_title('Palavras por segundo por COB', fontsize = 18)
    a.set_ylabel('Média de palavras por segundo', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'tonal_units', hue = 'corpus', ci = False, estimator =
np.mean)
    a.set_title('Unidades tonais por COB', fontsize = 18)
    a.set_ylabel('Média unidades tonais', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'number_of_words', hue = 'corpus', ci = False, estimator
= np.mean)
    a.set_title('Quantidade de palavras por COB', fontsize = 18)
    a.set_ylabel('Média palavras', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'ut_length', hue = 'corpus', ci = False, estimator =
np.mean)
    a.set_title('Duração das stanzas por COB', fontsize = 18)
```

```
    a.set_ylabel('Média de duração', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    padrao_enu_int = df_bal.query('inter_utterance > 0')
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = padrao_enu_int, x = 'COB', y = 'textual_units', hue = 'corpus', ci = False,
estimator = np.mean)
    a.set_title('Unidades textuais em enunciados interrompidos', fontsize = 18)
    a.set_ylabel('Média de Unidades Textuais', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(frameon=True, fontsize=13,  bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'words_per_second', hue = 'corpus', ci = False, estimator
= np.mean)
    a.set_title('Palavras por segundo por COB', fontsize = 18)
    a.set_ylabel('Média de palavras por segundo', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(loc="upper right", frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.78,0.5))
    plt.figure(figsize =(8, 5), dpi = 200)
    a = sns.barplot(data = df_bal, x = 'COB', y = 'AUX', hue = 'corpus', ci = False)
    a.set_title('Unidades Dialógicas por COB', fontsize = 18)
    a.set_ylabel('Média de Unidades Dialógicas', fontsize = 15)
    a.set_xlabel('COBs', fontsize = 15)
    a.tick_params(labelsize = 15)
    plt.xticks(rotation = 90)
    plt.legend(frameon=True, fontsize=13,  bbox_to_anchor = (0.5, 0.5,0.78,0.5))
try:
    if len(df_bal) > 0:
        from scipy.stats import mannwhitneyu
        if len(df_cob1_bal)> 0:
            lista_comparacao = ['COM', 'APC', 'CMM', 'TOP_TPL', 'APT', 'PAR_PRL', 'INT', 'SCA',
                'textual_units',                'tonal_units',                'AUX','inter_utterance',
'number_of_words','words_per_second',
                'retractings', 'retr_words']
```

```
        corpus_cesq_1COB= df_bal.query("corpus in 'Coral_esq' and COB == 1")
        corpus_cb_1COB = df_bal.query("corpus in 'Coral_brasil' and COB == 1")
        # corpus_shapiro_1COB = df_bal.query("COB in '1'")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_1COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_1COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_1_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_1_cob = pd.DataFrame([test_1_cob]).transpose()
        test_1_cob.reset_index(inplace=True)
        test_1_cob.columns = ['variavel', 'p_value']
        test_1_cob['p_value'] = test_1_cob['p_value']
        if len(test_1_cob) > 0:
            test_1_cob['COB'] = 1
            test_p_value= test_1_cob.copy()
            s_pvalue_1_cob = test_1_cob.query('p_value < 0.05')
    if len(df_cob2_bal) > 0:
        corpus_cesq_2COB= df_bal.query("corpus in 'Coral_esq' and COB == 2")
        corpus_cb_2COB = df_bal.query("corpus in 'Coral_brasil' and COB == 2")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_2COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_2COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_2_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_2_cob = pd.DataFrame([test_2_cob]).transpose()
        test_2_cob.reset_index(inplace=True)
        test_2_cob.columns = ['variavel', 'p_value']
        test_2_cob['p_value'] = test_2_cob['p_value']
        if len(test_2_cob) > 0:
            test_2_cob['COB'] = 2
```

```
            test_p_value = pd.concat([test_p_value, test_2_cob])
            s_pvalue_2_cob = test_2_cob.query('p_value < 0.05')
        if len(df_cob3_bal) > 0:
            corpus_cesq_3COB= df_bal.query("corpus in 'Coral_esq' and COB == 3")
            corpus_cb_3COB = df_bal.query("corpus in 'Coral_brasil' and COB == 3")
            lista_resultado = []
            lista_etiquetas = []
            for coluna_cb, linha_cb in corpus_cb_3COB.iteritems():
                for coluna_cesq, linha_cesq in corpus_cesq_3COB.iteritems():
                    if coluna_cb in lista_comparacao:
                        if coluna_cesq == coluna_cb:
                            lista_etiquetas.append(coluna_cesq)
                            lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
            test_3_cob = dict(zip(lista_etiquetas, lista_resultado))
            test_3_cob = pd.DataFrame([test_3_cob]).transpose()
            test_3_cob.reset_index(inplace=True)
            test_3_cob.columns = ['variavel', 'p_value']
            test_3_cob['p_value'] = test_3_cob['p_value']
            if len(test_3_cob) > 0:
                test_3_cob['COB'] = 3
                test_p_value = pd.concat([test_p_value, test_3_cob])
                s_pvalue_3_cob = test_3_cob.query('p_value < 0.05')
        if len(df_cob4_bal) > 0:
            corpus_cesq_4COB= df_bal.query("corpus in 'Coral_esq' and COB == 4")
            corpus_cb_4COB = df_bal.query("corpus in 'Coral_brasil' and COB == 4")
            lista_resultado = []
            lista_etiquetas = []
            for coluna_cb, linha_cb in corpus_cb_4COB.iteritems():
                for coluna_cesq, linha_cesq in corpus_cesq_4COB.iteritems():
                    if coluna_cb in lista_comparacao:
                        if coluna_cesq == coluna_cb:
                            lista_etiquetas.append(coluna_cesq)
                            lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
            test_4_cob = dict(zip(lista_etiquetas, lista_resultado))
            test_4_cob = pd.DataFrame([test_4_cob]).transpose()
            test_4_cob.reset_index(inplace=True)
            test_4_cob.columns = ['variavel', 'p_value']
            test_4_cob['p_value'] = test_4_cob['p_value']
```

```
        if len(test_4_cob) > 0:
            test_4_cob['COB'] = 4
            test_p_value = pd.concat([test_p_value, test_4_cob])
            s_pvalue_4_cob = test_4_cob.query('p_value < 0.05')
    if len(df_cob5_bal) > 0:
        corpus_cesq_5COB= df_bal.query("corpus in 'Coral_esq' and COB == 5")
        corpus_cb_5COB = df_bal.query("corpus in 'Coral_brasil' and COB == 5")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_5COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_5COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_5_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_5_cob = pd.DataFrame([test_5_cob]).transpose()
        test_5_cob.reset_index(inplace=True)
        test_5_cob.columns = ['variavel', 'p_value']
        test_5_cob['p_value'] = test_5_cob['p_value']
        if len(test_5_cob) > 0:
            test_5_cob['COB'] = 5
            test_p_value = pd.concat([test_p_value, test_5_cob])
            s_pvalue_5_cob = test_5_cob.query('p_value < 0.05')
    if len(df_cob6_bal) > 0:
        corpus_cesq_6COB= df_bal.query("corpus in 'Coral_esq' and COB == 6")
        corpus_cb_6COB = df_bal.query("corpus in 'Coral_brasil' and COB == 6")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_6COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_6COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_6_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_6_cob = pd.DataFrame([test_6_cob]).transpose()
        test_6_cob.reset_index(inplace=True)
```

```
        test_6_cob.columns = ['variavel', 'p_value']
        test_6_cob['p_value'] = test_6_cob['p_value']
        if len(test_6_cob) > 0:
            test_6_cob['COB'] = 6
            test_p_value = pd.concat([test_p_value, test_6_cob])
            s_pvalue_6_cob = test_6_cob.query('p_value < 0.05')
    if len(df_cob7_bal) > 0:
        corpus_cesq_7COB= df_bal.query("corpus in 'Coral_esq' and COB == 7")
        corpus_cb_7COB = df_bal.query("corpus in 'Coral_brasil' and COB == 7")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_7COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_7COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_7_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_7_cob = pd.DataFrame([test_7_cob]).transpose()
        test_7_cob.reset_index(inplace=True)
        test_7_cob.columns = ['variavel', 'p_value']
        test_7_cob['p_value'] = test_7_cob['p_value']
        if len(test_7_cob) > 0:
            test_7_cob['COB'] = 7
            test_p_value = pd.concat([test_p_value, test_7_cob])
            s_pvalue_7_cob = test_7_cob.query('p_value < 0.05')
    if len(df_cob8_bal) > 0:
        corpus_cesq_8COB= df_bal.query("corpus in 'Coral_esq' and COB == 8")
        corpus_cb_8COB = df_bal.query("corpus in 'Coral_brasil' and COB == 8")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_8COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_8COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_8_cob = dict(zip(lista_etiquetas, lista_resultado))
```

```
        test_8_cob = pd.DataFrame([test_8_cob]).transpose()
        test_8_cob.reset_index(inplace=True)
        test_8_cob.columns = ['variavel', 'p_value']
        test_8_cob['p_value'] = test_8_cob['p_value']
        if len(test_8_cob) > 0:
            test_8_cob['COB'] = 8
            test_p_value = pd.concat([test_p_value, test_8_cob])
            s_pvalue_8_cob = test_8_cob.query('p_value < 0.05')
    if len(df_cob9_bal) > 0:
        corpus_cesq_9COB= df_bal.query("corpus in 'Coral_esq' and COB == 9")
        corpus_cb_9COB = df_bal.query("corpus in 'Coral_brasil' and COB == 9")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_9COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_9COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_9_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_9_cob = pd.DataFrame([test_9_cob]).transpose()
        test_9_cob.reset_index(inplace=True)
        test_9_cob.columns = ['variavel', 'p_value']
        test_9_cob['p_value'] = test_9_cob['p_value']
        if len(test_9_cob) > 0:
            test_9_cob['COB'] = 9
            test_p_value = pd.concat([test_p_value, test_9_cob])
            s_pvalue_9_cob = test_9_cob.query('p_value < 0.05')
    if len(df_cob10_bal) > 0:
        corpus_cesq_10COB= df_bal.query("corpus in 'Coral_esq' and COB == 10")
        corpus_cb_10COB = df_bal.query("corpus in 'Coral_brasil' and COB == 10")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_10COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_10COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
```

```
                    lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_10_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_10_cob = pd.DataFrame([test_10_cob]).transpose()
        test_10_cob.reset_index(inplace=True)
        test_10_cob.columns = ['variavel', 'p_value']
        test_10_cob['p_value'] = test_10_cob['p_value']
        if len(test_10_cob) > 0:
            test_10_cob['COB'] = 10
            test_p_value = pd.concat([test_p_value, test_10_cob])
            s_pvalue_10_cob = test_10_cob.query('p_value < 0.05')
    if len(df_cob11_bal) > 0:
        corpus_cesq_11COB= df_bal.query("corpus in 'Coral_esq' and COB == 11")
        corpus_cb_11COB = df_bal.query("corpus in 'Coral_brasil' and COB == 11")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_11COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_11COB.iteritems():
                if coluna_cb in lista_comparacao:
                    if coluna_cesq == coluna_cb:
                        lista_etiquetas.append(coluna_cesq)
                        lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_11_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_11_cob = pd.DataFrame([test_11_cob]).transpose()
        test_11_cob.reset_index(inplace=True)
        test_11_cob.columns = ['variavel', 'p_value']
        test_11_cob['p_value'] = test_11_cob['p_value']
        if len(test_11_cob) > 0:
            test_11_cob['COB'] = 11
            test_p_value = pd.concat([test_p_value, test_11_cob])
            s_pvalue_11_cob = test_11_cob.query('p_value < 0.05')
    if len(df_cob12_bal) > 0:
        corpus_cesq_12COB= df_bal.query("corpus in 'Coral_esq' and COB == 12")
        corpus_cb_12COB = df_bal.query("corpus in 'Coral_brasil' and COB == 12")
        lista_resultado = []
        lista_etiquetas = []
        for coluna_cb, linha_cb in corpus_cb_12COB.iteritems():
            for coluna_cesq, linha_cesq in corpus_cesq_12COB.iteritems():
                if coluna_cb in lista_comparacao:
```

```
                if coluna_cesq == coluna_cb:
                    lista_etiquetas.append(coluna_cesq)
                    lista_resultado.append(mannwhitneyu(linha_cb, linha_cesq, alternative='two-
sided')[1])
        test_12_cob = dict(zip(lista_etiquetas, lista_resultado))
        test_12_cob = pd.DataFrame([test_12_cob]).transpose()
        test_12_cob.reset_index(inplace=True)
        test_12_cob.columns = ['variavel', 'p_value']
        test_12_cob['p_value'] = test_12_cob['p_value']
        if len(test_12_cob) > 0:
            test_12_cob['COB'] = 12
            test_p_value = pd.concat([test_p_value, test_12_cob])
            s_pvalue_12_cob = test_12_cob.query('p_value < 0.05')
    test_p_value.to_csv('test_p_value.csv')
    test_p_value.to_excel('test_p_value.xlsx')
except:
    pass
try:
    if len(test_p_value) > 0:
        test_p_value = test_p_value.query('p_value <= 0.05')
        plt.figure(figsize =(8, 5), dpi = 200)
        a = sns.barplot(data = test_p_value, x = 'COB', y = 'p_value', hue = 'variavel', palette = 'inferno')
        a.set_title('Resultados estatisticamente relevantes', fontsize = 18)
        a.set_ylabel('p-value - log', fontsize = 15)
        a.set_xlabel('COBs', fontsize = 15)
        a.tick_params(labelsize = 15)
        a.set_yscale("log")
        plt.xticks(rotation = 90)
        plt.legend(frameon=True, fontsize=13, bbox_to_anchor = (0.5, 0.5,0.85,0.5))
except:
    pass
if "=PAUSA_P=" in df.utterances.values:
    try:
        df['TMT'] = df['utterances'].apply(lambda x: len(re.findall(r"=PAUSA_P=?", x)))
        #df['corpus'] = df['audio'].apply(lambda x: "Coral_esq" if x.startswith('m') else 'Coral_brasil')
        df['word_before']= df['utterances_POS'].apply(lambda x: ' '.join(re.findall(r"(?<=\[)\('(\w+)", x)))
        df['class_before'] = df['utterances_POS'].apply(lambda x: ' '.join(re.findall(r"(?<=\[)\('\w+',\s'(\w+)",
x)))
        pausas_index = pd.Series((df.index[df['utterances'] == '=PAUSA_P=']))
        pausas_mais_um = pd.Series((df.index[df['utterances'] == '=PAUSA_P='] + 1))
```

```
        valores = df['ut_length'].iloc[pausas_index]
        df['pause_len'] = valores.set_axis(pausas_mais_um)
        df['pause_len'] = df['pause_len'].apply(lambda x: 0 if pd.isnull(x) == True else x)
        df_pauses = df.query('pause_len > 0')
        df_pauses = df_pauses.query('word_before != "PAUSA_P"')
        df_pauses['word_before'] = df_pauses['word_before'].apply(lambda x: '0' if len(x) < 1 else x)
        df_pauses['class_before'] = df_pauses['class_before'].apply(lambda x: '0' if len(x) < 1 else x)
        df_pauses = df_pauses.query('word_before != "0"')
        df_pauses = df_pauses.query('class_before != "0"')
        # df.to_csv('df_total.csv')
        # df_pauses.to_csv('df_pausas.csv')
        freq_dist = pd.DataFrame(df_pauses['word_before'].value_counts())
        freq_dist.reset_index(inplace=True)
        freq_dist.columns = ['palavra_diante_pausa', 'frequência']
        freq_dist.to_csv('df_palavras_diante.csv')
        sns.set_style('whitegrid')
        plt.figure(dpi = 300, figsize=(7, 5))
        a = sns.histplot(data= df_pauses, x = 'pause_len', kde = True)
        a.set_title('Distribuição da duração das pausas', fontsize = 16)
        a.set_xlabel("Tempo - (s)",fontsize= 14)
        a.set_ylabel("Frequência",fontsize = 15)
        a.tick_params(labelsize=15)
        plt.show()
        sns.set_style('whitegrid')
        plt.figure(dpi = 200, figsize=(8, 5))
        a = sns.barplot(data = df_pauses , x = 'class_before', y = 'pause_len',  palette = 'inferno', estimator
= np.mean, ci = False)
        a.set_title('Duração de pausas diante de classes de palavras', fontsize = 16)
        a.set_xlabel("Classes de palavras",fontsize= 14)
        a.set_ylabel("Média de duração",fontsize = 15)
        a.tick_params(labelsize=15)
        plt.xticks(rotation=90)
        plt.show()
        plt.figure(dpi = 200, figsize=(8, 5))
        a = sns.countplot(data = df_pauses, x = 'class_before', palette = 'inferno')
        a.set_title('Frequência de pausas diante de classes de palavras', fontsize = 16)
        a.set_xlabel("Classes de palavras",fontsize= 14)
        a.set_ylabel("Frequência",fontsize = 15)
        a.tick_params(labelsize=15)
        plt.xticks(rotation=90)
```

```
        plt.show()
        plt.figure(dpi = 200, figsize=(8, 5))
        a = sns.lineplot(data = freq_dist[:30], x = 'palavra_diante_pausa', y = 'frequência', palette = 'inferno')
        a.set_title('Palavras mais frequentes diante de pausas preenchidas', fontsize = 16)
        a.set_xlabel("Classes de palavras",fontsize= 14)
        a.set_ylabel("Frequência",fontsize = 15)
        a.tick_params(labelsize=15)
        plt.xticks(rotation=90)
        plt.show()
    except:
        pass
#daqui
try:
    if len(df_bal) > 0:
        if "Coral_brasil" and "Coral_esq" in df_bal.corpus.values:
            df_bal_cb = df_bal.query('corpus in "Coral_brasil"')
            dist_ret_cb = df_bal_cb['dist_retractings'].tolist()
            list_participants_cb = df_bal_cb['participant'].to_list()
            list_ret_cb= []
            for x in dist_ret_cb:
                for y in x.split():
                    list_ret_cb.append(y)
            list_ret_cb = '\n'.join(list_ret_cb)
            list_ret_cb = re.sub('\[|\]|\"|,', '', list_ret_cb)
            list_ret_cb = [x.lower() for x in list_ret_cb.splitlines() if len(x) > 0]
            list_ret_df_bal_cb = pd.DataFrame([x.strip() for x in list_ret_cb], columns=['retractings_full'])
            list_ret_df_bal_cb['corpus'] = 'Coral_brasil'
            #cesq
            df_bal_cesq = df_bal.query('corpus in "Coral_esq"')
            dist_ret_cesq = df_bal_cesq['dist_retractings'].tolist()
            list_participants_cesq = df_bal_cesq['participant'].to_list()
            list_ret_cesq= []
            for x in dist_ret_cesq:
                for y in x.split():
                    list_ret_cesq.append(y)
            list_ret_cesq = '\n'.join(list_ret_cesq)
            list_ret_cesq = re.sub('\[|\]|\"|,', '', list_ret_cesq)
            list_ret_cesq = [x.lower() for x in list_ret_cesq.splitlines() if len(x) > 0]
            list_ret_df_bal_cesq = pd.DataFrame([x.strip() for x in list_ret_cesq], columns=['retractings_full'])
            list_ret_df_bal_cesq['corpus'] = 'Coral_esq'
```

```
        list_ret_df_bal = pd.concat([list_ret_df_bal_cb, list_ret_df_bal_cesq])
    else:
        dist_ret = df_bal['dist_retractings'].tolist()
        list_participants = df_bal['participant'].to_list()
        list_ret= []
        for x in dist_ret:
            for y in x.split():
                list_ret.append(y)
        list_ret = '\n'.join(list_ret)
        list_ret = re.sub('\[|\]|\"|,', '', list_ret)
        list_ret = [x.lower() for x in list_ret.splitlines() if len(x) > 0]
        list_ret_df_bal = pd.DataFrame([x.strip() for x in list_ret], columns=['retractings_full'])
    list_ret_df_bal['retractings'] = list_ret_df_bal['retractings_full'].apply(lambda x: re.sub(r"&|-|<|>|\=",
'', x))
    list_ret_df_bal['retractings_syl'] = list_ret_df_bal['retractings'].apply(lambda x:
len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?a
u.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?",
str(x), flags = re.IGNORECASE)))
    list_ret_df_bal['oclus_des'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r"^p|^k|^k$|^c$|^ca|^co|^cu|^cã|^cô|^câ|^cõ|^cú|^q.?.?|^cr|^te(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|
b|d|g|v|z|j|m|n|u|ú|ó|o|ô|õ)|^té|^tê|^ta|^tá|^tã|^tâ|^to|^tô|^tu|^tú|^tr", x)))
    list_ret_df_bal['oclus_voz'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r"^b|^ga|^go|^gu|^gão|^gá|^gó|^gú|^gâ|^gr|^g$|^da|^dá|^dã|^dõ|^dô|^dú|^du|^do|^dr|^de
(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)", x)))
    list_ret_df_bal['laterais'] = list_ret_df_bal['retractings'].apply(lambda x: ' '.join(re.findall(r"(^l|^lh)",
x)))
    list_ret_df_bal['fricativas_des'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r"(^s|^x|^r|^ch|^f|^ce|^ci|^cê|^cí|^cé)", x)))
    list_ret_df_bal['fricativas_voz'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r'(^z|^j|^v|^ge|^gi)', x)))
    list_ret_df_bal['oclus_nas'] = list_ret_df_bal['retractings'].apply(lambda x: ' '.join(re.findall(r'(^m|^n)',
x)))
    list_ret_df_bal['africadas_des'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r'^te(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|a|á|ã|à|â|e|é|ê|i|í|o|ó|õ|ô|u|ú)|^ti|^t$', x)))
    list_ret_df_bal['africadas_voz'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r'^de(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)|^di|^d$', x)))
    list_ret_df_bal['vogais'] = list_ret_df_bal['retractings'].apply(lambda x: '
'.join(re.findall(r"^a|^e|^i|^o|^u|^ã|^õ|^à|^á|^â|^ô|^ó|^é|^ê|^ú|^h", x)))
    list_ret_df_bal = list_ret_df_bal.loc[list_ret_df_bal['retractings_full'] != 'xxx']
    list_ret_df_bal = list_ret_df_bal.loc[list_ret_df_bal['retractings_full'] != 'xxxx']
```

```
    list_ret_df_bal = list_ret_df_bal.loc[list_ret_df_bal['retractings_full'] != 'yyy']
    list_ret_df_bal = list_ret_df_bal.loc[list_ret_df_bal['retractings_full'] != 'yyyy']
    list_ret_df_bal = list_ret_df_bal.loc[list_ret_df_bal['retractings_full'] != 'hhh']
    list_ret_df_bal['retractings'] = list_ret_df_bal['retractings'].apply(lambda x: x[:3] if len(x) > 3 else x)
    list_ret_df_bal['oclus_des'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r"^p|^k|^k$|^c$|^ca|^co|^cu|^cã|^cô|^câ|^cõ|^cú|^q.?.?|^cr|^te(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|u|ú|ó|o|ô|õ)|^té|^tê|^ta|^tá|^tã|^tâ|^to|^tô|^tu|^tú|^tr", str(x))))
    list_ret_df_bal['oclus_voz'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r"^b|^ga|^go|^gu|^gão|^gá|^gó|^gú|^gâ|^gr|^g$|^da|^dá|^dã|^dõ|^dô|^dú|^du|^do|^dr|^de(?=p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)", x)))
    list_ret_df_bal['laterais'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r"(^l|^lh)", x)))
    list_ret_df_bal['fricativas_des'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r"(^s|^x|^r|^ch|^f|^ce|^ci|^cê|^cí|^cé)", x)))
    list_ret_df_bal['fricativas_voz'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r'(^z|^j|^v|^ge|^gi)', x)))
    list_ret_df_bal['oclus_nas'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r'(^m|^n)', x)))
    list_ret_df_bal['africadas_des'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r'^te(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|a|á|ã|à|â|e|é|ê|i|í|o|ó|õ|ô|u|ú)|^ti|^t$', x)))
    list_ret_df_bal['africadas_voz'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r'^de(?!p|t|k|q|c|f|s|ss|x|ch|l|r|rr|b|d|g|v|z|j|m|n|i|u|a|á|ã|â|à|e|é|ê|i|í|ó|ô|õ|u|ú)|^di|^d$', x)))
    list_ret_df_bal['vogais'] = list_ret_df_bal['retractings'].apply(lambda x: len(re.findall(r"^a|^e|^i|^o|^u|^ã|^õ|^à|^á|^â|^ô|^ó|^é|^ê|^ú|^h", x)))
    if "Coral_brasil" and "Coral_esq" in df_bal.corpus.values:
        retracted_words_bal = pd.DataFrame(list_ret_df_bal.groupby('corpus')['retractings_full'].value_counts())
        retracted_words_bal.columns = ['Frequência']
        retracted_words_bal.reset_index(inplace=True)
        retracted_words_bal.columns = ['corpus', 'retractings', 'Frequência']
    else:
        contagem_retratadas = pd.DataFrame(list_ret_df_bal['retractings_full'].value_counts())
        contagem_retratadas.reset_index(inplace=True)
        contagem_retratadas.columns = ['retractings', 'Frequência']
    if "Coral_brasil" and "Coral_esq" in df_bal.corpus.values:
        plt.figure(figsize =(9, 5), dpi = 200)
        a = sns.barplot(data = retracted_words.sort_values(by = 'Frequência', ascending =False)[:30], x = 'retractings', y = 'Frequência', hue = 'corpus')
        a.set_title('Palavras retratadas nos corpora', fontsize = 18)
        a.set_ylabel('Quantidade', fontsize = 15)
        a.set_xlabel('Classe do segmento', fontsize = 15)
```

```
        a.tick_params(labelsize = 15)
        plt.xticks(rotation = 90)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
    else:
        plt.figure(figsize =(9, 5), dpi = 200)
        a = sns.barplot(data = contagem_retratadas[:30], x = 'retractings', y = 'Frequência')
        a.set_title('Frequência de palavras retratadas', fontsize = 18)
        a.set_ylabel('Quantidade', fontsize = 15)
        a.set_xlabel('Classe do segmento', fontsize = 15)
        a.tick_params(labelsize = 15)
        plt.xticks(rotation = 90)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
    if "Coral_brasil" and "Coral_esq" in df_bal.corpus.values:
        list_ret_df_bal_visu_cb = list_ret_df_bal.query('corpus in "Coral_brasil"')
        list_ret_df_bal_visu_cb.drop(list_ret_df_bal_visu_cb.loc[:, 'retractings_full': 'retractings_syl'],
axis = 1, inplace=True)
        # list_ret_df_bal_visu = list_ret_df_bal_visu_cb.loc[:, 'oclus_des':]
        # list_ret_df_bal_visu_cb.drop('corpus', axis =1, inplace= True)
        list_ret_df_bal_visu_cb = list_ret_df_bal_visu_cb.melt()
        list_ret_df_bal_visu_cb =
pd.DataFrame(list_ret_df_bal_visu_cb.groupby('variable')['value'].sum())
        list_ret_df_bal_visu_cb.reset_index(inplace=True)
        list_ret_df_bal_visu_cb['corpus'] = 'Coral_brasil'
        #cesq
        list_ret_df_bal_visu_cesq = list_ret_df_bal.query('corpus in "Coral_esq"')
        list_ret_df_bal_visu_cesq.drop(list_ret_df_bal_visu_cesq.loc[:, 'retractings_full':
'retractings_syl'], axis = 1, inplace=True)
        # list_ret_df_bal_visu = list_ret_df_bal_visu_cesq.loc[:, 'oclus_des':]
        # list_ret_df_bal_visu_cesq.drop('corpus', axis =1, inplace= True)
        list_ret_df_bal_visu_cesq = list_ret_df_bal_visu_cesq.melt()
        list_ret_df_bal_visu_cesq =
pd.DataFrame(list_ret_df_bal_visu_cesq.groupby('variable')['value'].sum())
        list_ret_df_bal_visu_cesq.reset_index(inplace=True)
        list_ret_df_bal_visu_cesq['corpus'] = 'Coral_esq'
        list_ret_df_bal_classe = pd.concat([list_ret_df_bal_visu_cb, list_ret_df_bal_visu_cesq],
ignore_index=True)
        list_ret_df_bal_classe.sort_values(by ='value', ascending = False, inplace=True)
        list_ret_df_bal_classe.columns = ['Classe_do_segmento', 'Frequência', 'corpus']
        sns.set_style('whitegrid')
        plt.figure(figsize =(7, 5), dpi = 200)
```

```
        a = sns.barplot(data = list_ret_df_bal_classe,x = 'Classe_do_segmento', y = 'Frequência', hue = 'corpus', estimator = sum )
        a.set_title('Classe do segmento inicial do retracting por corpora', fontsize = 18)
        a.set_ylabel('Quantidade', fontsize = 15)
        a.set_xlabel('Classe do segmento', fontsize = 15)
        a.tick_params(labelsize = 15)
        plt.xticks(rotation = 90)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
    else:
        list_ret_df_bal_visu = list_ret_df_bal.copy()
        list_ret_df_bal_visu.drop(list_ret_df_bal_visu.loc[:, 'retractings_full': 'retractings_syl'], axis = 1, inplace=True)
        # list_ret_df_bal_visu = list_ret_df_bal_visu.loc[:, 'oclus_des':]
        # list_ret_df_bal_visu.drop('corpus', axis =1, inplace= True)
        list_ret_df_bal_visu = list_ret_df_bal_visu.melt()
        list_ret_df_bal_visu = pd.DataFrame(list_ret_df_bal_visu.groupby('variable')['value'].sum())
        list_ret_df_bal_visu.reset_index(inplace=True)
        # list_ret_df_bal_visu['corpus'] = 'Coral_brasil'
        list_ret_df_bal_visu.columns = ['Classe_do_segmento', 'Frequência']
        sns.set_style('whitegrid')
        plt.figure(figsize =(7, 5), dpi = 200)
        a = sns.barplot(data = list_ret_df_bal_visu.sort_values(by = 'Frequência', ascending = False),x = 'Classe_do_segmento', y = 'Frequência', palette = 'inferno')
        a.set_title('Classe do segmento inicial do retracting', fontsize = 18)
        a.set_ylabel('Quantidade', fontsize = 15)
        a.set_xlabel('Classe do segmento', fontsize = 15)
        a.tick_params(labelsize = 15)
        plt.xticks(rotation = 90)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
except:
    pass
if "Coral_brasil" and "Coral_esq" in df_bal.corpus.values:
    if df_bal.textual_units.sum() > 0:
        counted_patterns = pd.DataFrame(df_bal.groupby('corpus')['filtered_patterns'].value_counts())
        counted_patterns.columns = ['Frequência']
        counted_patterns.reset_index(inplace=True)
        counted_patterns.columns = ['corpus', 'Padrões_inf', 'Frequência']
        counted_patterns['Padrões_inf'] = counted_patterns['Padrões_inf'].str.replace('=', '').str.strip()
        counted_patterns['Padrões_inf']= counted_patterns['Padrões_inf'].apply(lambda x: '0' if len(x) < 3 else x)
```

```
        counted_patterns = counted_patterns.query('Padrões_inf != "0"')
        sns.set_style('whitegrid')
        plt.figure(dpi = 200, figsize = (9, 5))
        plt.xticks(rotation=90)
        b = sns.barplot(data = counted_patterns.sort_values(by = 'Frequência', ascending= False)[:15], x
= 'Padrões_inf', y = "Frequência", hue = 'corpus')
        b.set_title('Padrões informacionais mais frequentes nas amostras balanceadas', fontsize = 16)
        b.set_ylabel("Frequência", fontsize = 15)
        b.set_xlabel('Padrões informacionais',fontsize=14)
        b.tick_params(labelsize=15)
        plt.legend(loc="upper right", frameon=True, fontsize=13)
if "Coral_brasil" and "Coral_esq" in df.corpus.values:
    df.to_csv('df_all_data.csv')
    df_bal.to_csv('df_bal.csv')
    df_speech_m.to_csv('df_speech_m.csv')
    df_inform_m.to_csv('df_textual_units_per_participant.csv')
    grouped_filt_patterns.to_csv('df_patterns_participants.csv')
    list_ret_df_bal.to_csv('df_retractings_bal.csv')
    retracted_words_bal.to_csv('df_retracted_words_count_bal.csv')
    retracted_words.to_csv('df_retracted_words_count_all.csv')
else:
    df.to_csv('df_all_data.csv')
    df_speech_m.to_csv('df_speech_m.csv')
    df_inform_m.to_csv('df_textual_units_per_participant.csv')
    grouped_filt_patterns.to_csv('df_patterns_participants.csv')
    list_ret_df.to_csv('df_retractings.csv')
    contagem_retratadas.to_csv('df_retracted_words_count.csv')
print('Let me know if I can help you with something! \n by: José Carlos Costa \n
email:carlosjuniorcosta1@gmail.com')
```

Anexo B - Programa 2: Descritor lexical para C-ORAL-BRASIL e C-ORAL-ESQ

Link para o programa: https://colab.research.google.com/drive/1-NZ3krrWiHUmCJTMgtnRqEjYnAzLg5nR?usp=sharing

Link para atualizações: https://github.com/carlosjuniorcosta1/extrator_2_coral_esq_coral_brasil

Finalidade: Contabiliza as classes gramaticais e oferece detalhes morfológicos das classes lexicais por meio da integração do DELAF e bibliotecas como NLTK e Spacy, do Python.

Instruções de uso

Tutorial em vídeo 1 - Arquivos requisitados e passo a passo para usar o Programa 2: https://www.loom.com/share/d506efde2b7e4f268f777a07976f7d3b

Tutorial em vídeo 2 – Resultados do Programa 2: https://www.loom.com/share/f8777b37592b436d8687b00f85fbdd6c

Fazer o upload do DELAF e do dataframe obtido no Programa 1, de extração de padrões informacionais. Após isso, clicar no executar, aguardar a instalação e reiniciar o ambiente de execução pelo menu[49]. Posteriormente, clicar na célula e executar novamente. Caso o usuário queira ver sua nuvem de palavras (ver a seguir), que mostra de forma bastante diferente as palavras mais frequentes da transcrição em questão – ou do conjunto de transcrições utilizados- em forma de alguma figura[50], é possível fazer upload de uma imagem, em formato png, que vai ser automaticamente lida pelo programa e transformada em nuvem (ver Figuras 78 e 79). Para isso ocorrer corretamente, nomeie a figura como “figura” e tenha especial atenção ao esquema de cores de sua figura. É importante que o contorno que você modelar seja o mais escuro – preto, de preferência = possível e com os arredores mais claros possíveis – branco, de preferência (cf. exemplo para o áudio bfammn01, do C-ORAL-BRASIL, a seguir).

[49]No Spyder, essa operação de reiniciar não é necessária.
[50] O fundo deve ser branco e a figura o mais escura possível, como uma silhueta ou sombra.

Arquivos de entrada

Dataframe principal obtido no Programa 1 (o completo, com todos os participantes ou o personalizado, com o nome do participante filtrado);

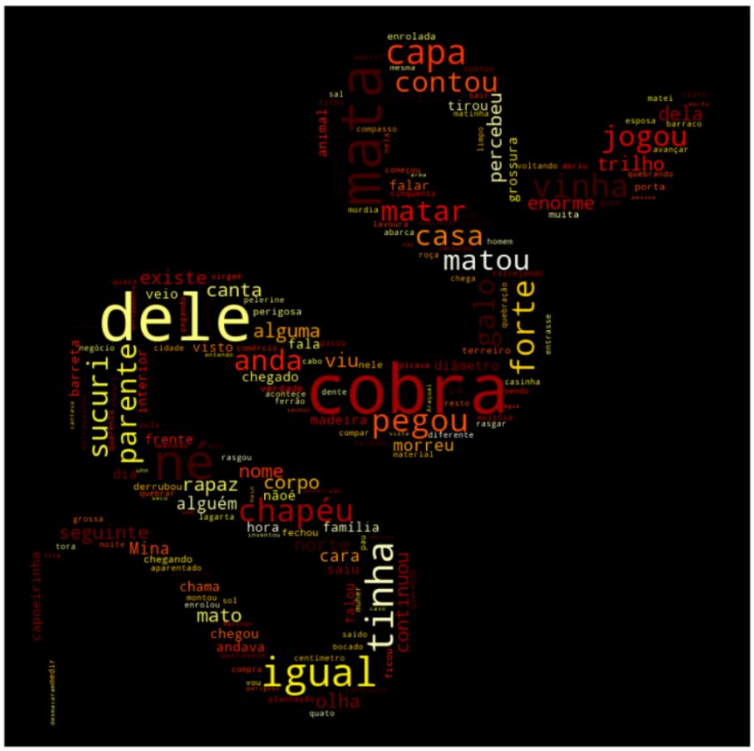
DELAF em txt utf-8[51]. Utilizei a versão 2.

Opcional: figura que o usuário quer que sua nuvem de palavras se transforme, em formato png, com fundo o mais branco possível e figura mais preta possível.

Figura 78 – Exemplo de figura para nuvem de palavras modelável para o áudio bfammn01, do C-ORAL-BRASIL.

Figura 79 – Nuvem de palavras moldada a partir da figura 78.

[51]http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html



Fonte: Elaboração do autor.

Arquivos de saída

Distribuição por frequência de todas as classes gramaticais;

Palavras mais frequentes na transcrição, com e sem stopwords;

Lema de substantivos, verbos, verbos auxiliares e adjetivos;

Flexões de classes nominais e conjugações verbais;

Quantidade de sílabas de verbos, substantivos, adjetivos e advérbios;

Gráficos plotados na tela e em png para download;

Gráficos de linha com a contagem de: adjetivos, substantivos, verbos auxiliares, verbos, tempo e modo verbal mais frequente;

Código:

```
#author: José Carlos Costa
#email: carlosjuniorcosta1@gmail.com

!pip install nltk
!pip install spacy
!python3 -m spacy download pt
import os
```

```
import pandas as pd
import spacy
import nltk
import numpy as np
from nltk import FreqDist
import re
from spacy.lang.pt.stop_words import STOP_WORDS
from string import punctuation
nltk.download('rslp')
stemmer = nltk.stem.RSLPStemmer()
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from PIL import Image
nlp = spacy.load("pt_core_news_sm")
file = input(str('Enter the filename'))
file_plot = file[:-4]
def count_pos(pos_column, pos_name):
    pos_column_list = []
    for sublist in pos_column:
        for item in sublist:
            pos_column_list.append(item)
    pos_flat_list = ' '.join(pos_column_list)
    pos_list_count = FreqDist(pos_flat_list.split())
    df_count = pd.DataFrame.from_dict([pos_list_count]).transpose().reset_index()
    df_count.columns = [pos_name, 'frequencia']
    df_count = df_count.sort_values(by='frequencia', ascending = False)
    return df_count
df = pd.read_csv(file)
df2 = pd.DataFrame(df['utterances_POS'])
df2.reset_index(inplace=True)
print('Contando distribuição por classes de palavras')
df2['adjetivos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='ADJ')", str(x)))
df2['adverbios'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='ADV')", str(x)))
df2['adverbios_cs'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='ADV-KS')", str(x)))
df2['adverbios_cs_rel'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='ADV-KS-REL')",
str(x)))
df2['artigos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='ART')", str(x)))
df2['conjuncoes_coordenadas'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='KC')",
str(x)))
```

```
df2['conjuncoes_subordinadas'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='KS')", str(x)))
df2['interjeicoes'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='IN')", str(x)))
df2['nomes_proprios'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='NPROP')", str(x)))
df2['numerais'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='NUM')", str(x)))
df2['substantivos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='N')", str(x)))
df2['participios_passado'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PCP')", str(x)))
df2['palavras_denotativas'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PDEN')", str(x)))
df2['preposicoes'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PREP')", str(x)))
df2['preposicoes_plus'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PREP\\|\\+')", str(x)))
df2['pronomes_adjetivos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PROADJ')", str(x)))
df2['pronomes_pessoais'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PROPESS')", str(x)))
df2['pronomes_substantivos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PROSUB')", str(x)))
df2['pronomes_cs'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PRO-KS')", str(x)))
df2['pro_cs_rel'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='PRO-KS-REL')", str(x)))
df2['verbos'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='V')", str(x)))
df2['verbos_aux'] = df2['utterances_POS'].apply(lambda x: re.findall("'(\w+)',\s(?='VAUX')", str(x)))
df2.to_csv('distribuicao_lexical.csv')
print('Gerando arquivos em csv')
adjetivos = count_pos(df2['adjetivos'].tolist(), 'adjetivos')
# adjetivos.to_csv(f'{file[:-4]}_adjetivos_dist.csv')
adverbios = count_pos(df2['adverbios'].tolist(), 'adverbios')
adverbios.to_csv(f'{file[:-4]}_adverbios_dist.csv')
# adverbios_cs = count_pos(df2['adverbios_cs'].tolist(), 'adverbios_cs')
# adverbios_cs.to_csv(f'{file[:-4]}_adverbios_cs_dist.csv')
# adverbios_cs_rel = count_pos(df2['adverbios_cs_rel'].tolist(), 'adverbios_cs_rel')
# adverbios_cs_rel.to_csv(f'{file[:-4]}_adverbios_cs_rel_dist.csv')
artigos = count_pos(df2['artigos'].tolist(), 'artigos')
artigos.to_csv(f'{file[:-4]}_artigos_dist.csv')
conjuncoes_coordenadas = count_pos(df2['conjuncoes_coordenadas'].tolist(), 'conjuncoes_coordenadas')
conjuncoes_coordenadas.to_csv(f'{file[:-4]}_conjuncoes_coordenadas_dist.csv')
```

```
conjuncoes_subordinadas = count_pos(df2['conjuncoes_subordinadas'].tolist(),
'conjuncoes_subordinadas')
conjuncoes_subordinadas.to_csv(f'{file[:-4]}_conjuncoes_subordinadas_dist.csv')
interjeicoes = count_pos(df2['interjeicoes'].tolist(), 'interjeicoes')
interjeicoes.to_csv(f'{file[:-4]}_interjeicoes_dist.csv')
nomes_proprios = count_pos(df2['nomes_proprios'].tolist(), 'nomes_proprios')
nomes_proprios.to_csv(f'{file[:-4]}_nomes_proprios_dist.csv')
numerais = count_pos(df2['numerais'].tolist(), 'numerais')
numerais.to_csv(f'{file[:-4]}_numerais_dist.csv')
substantivos = count_pos(df2['substantivos'].tolist(), 'substantivos')
# substantivos.to_csv(f'{file[:-4]}_substantivos_dist.csv')
participios_passado = count_pos(df2['participios_passado'].tolist(), 'participios_passado')
participios_passado.to_csv(f'{file[:-4]}_participios_passado_dist.csv')
palavras_denotativas = count_pos(df2['palavras_denotativas'].tolist(), 'palavras_denotativas')
palavras_denotativas.to_csv(f'{file[:-4]}_palavras_denotativas_dist.csv')
preposicoes = count_pos(df2['preposicoes'].tolist(), 'preposicoes')
preposicoes.to_csv(f'{file[:-4]}_preposicoes_dist.csv')
preposicoes_plus = count_pos(df2['preposicoes_plus'].tolist(), 'preposicoes_plus')
preposicoes_plus.to_csv(f'{file[:-4]}_preposicoes_plus_dist.csv')
pronomes_adjetivos = count_pos(df2['pronomes_adjetivos'].tolist(), 'pronomes_adjetivos')
pronomes_adjetivos.to_csv(f'{file[:-4]}_pronomes_adjetivos_dist.csv')
pronomes_pessoais = count_pos(df2['pronomes_pessoais'].tolist(), 'pronomes_pessoais')
pronomes_pessoais.to_csv(f'{file[:-4]}_pronomes_pessoais_dist.csv')
pronomes_substantivos = count_pos(df2['pronomes_substantivos'].tolist(), 'pronomes_substantivos')
pronomes_substantivos.to_csv(f'{file[:-4]}_pronomes_substantivos_dist.csv')
pronomes_cs = count_pos(df2['pronomes_cs'].tolist(), 'pronomes_cs')
pronomes_cs.to_csv(f'{file[:-4]}_pronomes_cs_dist.csv')
pro_cs_rel = count_pos(df2['pro_cs_rel'].tolist(), 'pro_cs_rel')
pro_cs_rel.to_csv(f'{file[:-4]}_pro_cs_rel_dist.csv')
verbos = count_pos(df2['verbos'].tolist(), 'verbos')
# verbos.to_csv(f'{file[:-4]}_verbos_dist.csv')
verbos_aux = count_pos(df2['verbos_aux'].tolist(), 'verbos_aux')
# verbos_aux.to_csv(f'{file[:-4]}_verbos_aux_dist.csv')
print('Arquivos em csv gerados! Aguarde pela verificação do lema e detalhes \n \
    morfológicos do Nilc')
#lematização e detalhamento morfológico com o Nilc
print('Verificando adjetivos')
adjetivos['adjetivos'] = adjetivos['adjetivos'].apply(lambda x: x.lower())
# adjetivos.columns = ['Unnamed: 0', 'adjetivos', 'frequencia', 'lema_spacy', 'raiz']
adjetivos_b = adjetivos['adjetivos'].tolist()
```

```
with open('DELAF_pb.txt', 'r', encoding='utf-8') as file2:
    nilc = file2.read()
    nilc = re.sub(r'\w.+\.(?!A:).+', '', nilc)
    nilc = re.sub(r'(?<=:)(D|A|S)(.+)', r'\1 \2', nilc)
    nilc = re.sub(r'(?<=.)A', '', nilc)
    #nilc = re.sub(r'\w+(?=\.)', '', nilc)
    nilc = re.sub(r'\.', '', nilc)
    nilc = re.sub(r',|:', ' ', nilc)
    nilc = [x for x in nilc.splitlines() if len(x)]
    nilc = [x.split() for x in nilc]
# adjetivos['adjetivos'] = adjetivos['adjetivos'].apply(lambda x: x.lower())
# adjetivos['lema'] = adjetivos['lema'].apply(lambda x: x.lower())
#verbos_aux_b = verbos_aux['verbos_aux'].tolist()
print('Conferindo lemas e detalhes morfológicos')
lista_adjetivos = []
lista_lemas_adj = []
for x in adjetivos_b:
    #print(palavra_out)
    for y in nilc:
        #print(x, y[0])
        if x == y[0]:
            lista_lemas_adj.append(y[1])
            lista_adjetivos.append((x, y[2:]))
df_adjetivos = pd.DataFrame(lista_adjetivos, columns=['adjetivos', 'flexao'])
df_adjetivos['lema_nilc'] = lista_lemas_adj
print('Lematizando com Spacy')
df_adjetivos['lema_spacy'] = df_adjetivos['adjetivos'].apply(lambda x: ' '.join([token.lemma_ for token in
nlp(x)]))
print('Encontrando raiz morfológica de adjetivos com o NLTK')
df_adjetivos['raiz_nltk'] = df_adjetivos['adjetivos'].apply(lambda x: stemmer.stem(x))
df_adjetivos = adjetivos.merge(df_adjetivos, on='adjetivos', how='outer')
df_adjetivos_f = df_adjetivos[pd.notnull(df_adjetivos['flexao'])]
df_adjetivos_f['femininos'] = df_adjetivos_f['flexao'].apply(lambda x: re.findall(r'.(f.)', str(x)))
df_adjetivos_f['masculinos'] = df_adjetivos_f['flexao'].apply(lambda x: re.findall(r'.(m.)', str(x)))
df_adjetivos_f['fem_cont_sing'] = df_adjetivos_f['femininos'].apply(lambda x: len(re.findall(r'.(fs)',
str(x))))
df_adjetivos_f['fem_cont_pl'] = df_adjetivos_f['femininos'].apply(lambda x: len(re.findall(r'.(fp)', str(x))))
df_adjetivos_f['masc_cont_sing'] = df_adjetivos_f['masculinos'].apply(lambda x: len(re.findall(r'.(ms)',
str(x))))
```

```
df_adjetivos_f['masc_cont_pl'] = df_adjetivos_f['masculinos'].apply(lambda x: len(re.findall(r'.(mp)', str(x))))
df_adjetivos_f['syl_size'] = df_adjetivos_f['adjetivos'].apply(lambda x: len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?au.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?", str(x), flags = re.IGNORECASE)))
df_adjetivos_f.to_csv('adjetivos_dist.csv')
print('Plotando um lineplot de distribuição de adjetivos mais frequentes')
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = df_adjetivos.sort_values(by='frequencia', ascending = False)[:30], x= 'adjetivos', y = 'frequencia', marker = 's',  lw='3', color = 'orange')
b.set_title(f'Adjetivos mais frequentes em {file_plot}', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Adjetivos',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig(f'{file_plot}_adjetivos_lineplot.png', dpi=300)
#substantivos
print('Verificando os substantivos no Nilc')
substantivos['substantivos'] = substantivos['substantivos'].apply(lambda x: x.lower())
# substantivos.columns = ['Unnamed: 0', 'substantivos', 'frequencia', 'lema_spacy', 'raiz']
with open('DELAF_pb.txt', 'r', encoding='utf-8') as file2:
    nilc = file2.read()
    nilc = re.sub(r'\w.+\.(?!N).+', '', nilc)
    nilc = re.sub(r'(?<=:)(D|A)(.+)', r'\1 \2', nilc)
    #nilc = re.sub(r'\w+(?=\.)', '', nilc)
    nilc = re.sub(r'\.', ',', nilc)
    nilc = re.sub(r':', ',', nilc)
    # nilc = re.sub(r'N(?=\s)', '', nilc)
    nilc = [x for x in nilc.splitlines() if len(x)]
    nilc = [x.split(',') for x in nilc]
substantivos_b = substantivos['substantivos'].tolist()
print('Verificando lemas e detalhes morfológicos de substantivos')
lista_nomes = []
lista_lemas_nomes = []
for x in substantivos_b:
    #print(palavra_out)
    for y in nilc:
        #print(x, y[1:])
```

```
        if x == y[0]:
            lista_nomes.append((x, y[2:]))
            lista_lemas_nomes.append(y[1])
df_nomes = pd.DataFrame(lista_nomes, columns=['substantivos', 'flexao'])
df_nomes['lema_nilc'] = lista_lemas_nomes
print('Lematizando substantivos com Spacy')
df_nomes['lema_spacy'] = df_nomes['substantivos'].apply(lambda x: ' '.join([token.lemma_ for token in nlp(x)]))
print('Inserindo raiz morfológica com NLTK')
df_nomes['raiz_nltk'] = df_nomes['substantivos'].apply(lambda x: stemmer.stem(x))
df_nomes = substantivos.merge(df_nomes, on='substantivos', how='outer')
df_substantivos_f = df_nomes[pd.notnull(df_nomes['flexao'])]
df_substantivos_f['femininos'] = df_substantivos_f['flexao'].apply(lambda x: re.findall(r'.(f.)', str(x)))
df_substantivos_f['masculinos'] = df_substantivos_f['flexao'].apply(lambda x: re.findall(r'.(m.)', str(x)))
df_substantivos_f['fem_cont_sing'] = df_substantivos_f['femininos'].apply(lambda x: len(re.findall(r'.(fs)', str(x))))
df_substantivos_f['fem_cont_pl'] = df_substantivos_f['femininos'].apply(lambda x: len(re.findall(r'.(fp)', str(x))))
df_substantivos_f['masc_cont_sing'] = df_substantivos_f['masculinos'].apply(lambda x: len(re.findall(r'.(ms)', str(x))))
df_substantivos_f['masc_cont_pl'] = df_substantivos_f['masculinos'].apply(lambda x: len(re.findall(r'.(mp)', str(x))))
df_substantivos_f['syl_size'] = df_substantivos_f['substantivos'].apply(lambda x: len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?au.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?", str(x), flags = re.IGNORECASE)))
df_substantivos_f = df_substantivos_f.query('substantivos != "tipo"')
df_substantivos_f.to_csv(f'{file_plot}_substantivos_dist.csv')
#grafico substantivos
# df_substantivos = pd.read_csv('df_substantivos_f.csv')
# df_substantivos.drop(13, axis = 0, inplace = True)
print('Plotando lineplot com distribuição dos substantivos mais frequentes')
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = df_substantivos_f.sort_values(by='frequencia', ascending = False)[:30], x= 'substantivos', y = 'frequencia', marker = 's',  lw='3')
b.set_title(f'Substantivos mais frequentes em {file_plot}', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Substantivos',fontsize=15)
```

```
b.tick_params(labelsize=16)
plt.savefig('substantivos_lineplot.png', dpi=300)
#verbos aux
print('Verificando as entradas de verbos auxiliares no Nilc')
with open('DELAF_pb.txt', 'r', encoding='utf-8') as file2:
    nilc = file2.read()
    nilc = re.sub(r'\w.+\.(?!V:).+', '', nilc)
    nilc = re.sub(r'(?<=:)([A-Z])(\d\w)', r'\1 \2', nilc) #separa tempo da flexão de pessoa
    nilc = re.sub(r'(?<=.)V', '', nilc) # apaga o V da classe
    # nilc = re.sub(r'\w+(?=\.)', '', nilc) #tira o lema
    nilc = re.sub(r'\.|:|,', ' ', nilc)
    nilc = [x for x in nilc.splitlines() if len(x)]
    nilc = [x.split() for x in nilc]
verbos_aux['verbos_aux'] = verbos_aux['verbos_aux'].apply(lambda x: x.lower())
verbos_aux_b = verbos_aux['verbos_aux'].tolist()
print('Verificando lemas e detalhes morfológicos de verbos auxiliares')
lista_verbos_aux = []
lista_lemas_vaux = []
for x in verbos_aux_b:
    #print(palavra_out)
    for y in nilc:
        #print(x, y[0])
        if x == y[0]:
            lista_lemas_vaux.append(y[1])
            lista_verbos_aux.append((x, y[2:]))
df_vaux = pd.DataFrame(lista_verbos_aux)
df_vaux.columns = ['verbos_aux', 'classificacao_nilc']
df_vaux['lema_nilc'] = lista_lemas_vaux
print('Lematizando com Spacy')
df_vaux['lema_spacy'] = df_vaux['verbos_aux'].apply(lambda x: ' '.join([token.lemma_ for token in
nlp(x)]))
print('Inserindo raiz morfológica do verbo auxiliar com NLTK')
df_vaux['raiz_nltk'] = df_vaux['verbos_aux'].apply(lambda x: stemmer.stem(x))
lista_class = df_vaux['classificacao_nilc'].apply(lambda x: str(x)).tolist()
lista_class = '\n'.join(lista_class)
print('Inserindo detalhes morfológicos dos verbos auxiliares')
lista_class = re.sub(r"'W'", "'Infinitivo'", lista_class)
lista_class = re.sub("'G'", "'Gerúndio'", lista_class)
lista_class = re.sub(r"'K'", "'Particípio'", lista_class)
lista_class = re.sub(r"'I'", "'Pretérito Imperfeito do Indicativo'", lista_class)
```

```
lista_class= re.sub(r"'J'", "'Pretérito Perfeito do Indicativo'", lista_class)
lista_class = re.sub(r"'F'", "'Futuro do Presente do Indicativo'", lista_class)
lista_class = re.sub(r"'Q'", "'Pretérito mais que Perfeito do Indicativo'", lista_class)
lista_class = re.sub(r"'S'", "'Presente do Subjuntivo'", lista_class)
lista_class = re.sub(r"'T'", "'Imperfeito do Subjuntivo'", lista_class)
lista_class = re.sub(r"'U'", "'Futuro do Subjuntivo'", lista_class)
lista_class = re.sub(r"'Y'", "'Imperativo'", lista_class)
lista_class = re.sub(r"'C'", "'Futuro do Pretérito'", lista_class)
lista_class = re.sub(r"'P'", "'Presente do Indicativo'", lista_class)
df_vaux['classificacao_verbal'] = [x for x in lista_class.splitlines()]
df_vaux = verbos_aux.merge(df_vaux, on = 'verbos_aux')
df_vaux = df_vaux.query('lema_nilc != "abraçar" and lema_nilc != "acabar" and lema_nilc != "achar" and
lema_nilc != "voltar" and lema_nilc != "seriar" and lema_nilc != "continuar" and lema_nilc != "começar"
and lema_nilc != "podar" and lema_nilc != "estivar" and lema_nilc != "estudar" and lema_nilc != "olhar"
and lema_nilc != "saber" and lema_nilc != "ver" and lema_nilc != "vir" and lema_nilc != "dizer" and
lema_nilc != "fazer"')
df_vaux['syl_size'] = df_vaux['verbos_aux'].apply(lambda x:
len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?a
u.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?",
str(x), flags = re.IGNORECASE)))
df_vaux['classificacao_clean'] = df_vaux['classificacao_verbal'].apply(lambda x: re.sub(r'.\d+\w.', '', x))
df_vaux['classificacao_clean'] = df_vaux['classificacao_clean'].apply(lambda x: re.sub(r"\s,\s(?=')", '', x))
df_vaux['classificacao_clean'] = df_vaux['classificacao_clean'].apply(lambda x: re.sub(r",(?=\s$)", '', x))
df_vaux['infinitivo'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Infinitivo', x)))
df_vaux['gerundio'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Gerúndio', x)))
df_vaux['participio'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Particípio', x)))
df_vaux['pret_perf_ind'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito
Perfeito do Indicativo', x)))
df_vaux['pret_imp_ind'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito
Imperfeito do Indicativo', x)))
df_vaux['futuro_pret_ind'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do
Pretérito', x)))
df_vaux['pres_indicativo'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Presente do
Indicativo', x)))
df_vaux['futuro_pres_ind'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do
Presente do Indicativo', x)))
df_vaux['pret_mais_perf_ind'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito
mais que Perfeito do Indicativo', x)))
df_vaux['pres_subj'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Presente do
Subjuntivo', x)))
```

```
df_vaux['imp_subj'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Imperfeito do Subjuntivo', x)))
df_vaux['imperativo'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Imperativo', x)))
df_vaux['futuro_subj'] = df_vaux['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do Subjuntivo', x)))
df_vaux.drop('classificacao_clean', axis = 1, inplace = True)
df_vaux.to_csv(f'{file_plot}_verbos_aux_dist.csv')
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = df_vaux.sort_values(by='frequencia', ascending = False)[:30], x= 'verbos_aux', y = 'frequencia', marker = 's',  lw='3')
b.set_title(f'Verbos auxiliares mais frequentes em {file_plot}', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Verbos auxiliares',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig(f'{file_plot}_verbos_aux_lineplot.png', dpi=300)
#verbos
print('Conferindo as entradas verbais de verbos no Nilc')
with open('DELAF_pb.txt', 'r', encoding='utf-8') as file2:
    nilc = file2.read()
    nilc = re.sub(r'\w.+\.(?!V:).+', '', nilc)
    nilc = re.sub(r'(?<=:)([A-Z])(\d\w)', r'\1 \2', nilc) #separa tempo da flexão de pessoa
    nilc = re.sub(r'(?<=.)V', '', nilc) # apaga o V da classe
    # nilc = re.sub(r'\w+(?=\.)', '', nilc) #tira o lema
    nilc = re.sub(r'\.|:|,', ' ', nilc)
    nilc = [x for x in nilc.splitlines() if len(x)]
    nilc = [x.split() for x in nilc]
verbos_b = verbos['verbos'].tolist()
print('Conferindo lemas e detalhes morfológicos dos verbos no Nilc')
lista_verbos= []
lista_lemas_verbos = []
for x in verbos_b:
    #print(palavra_out)
    for y in nilc:
        #print(x, y[0])
        if x == y[0]:
            lista_lemas_verbos.append(y[1])
            lista_verbos.append((x, y[2:]))
df_verbos = pd.DataFrame(lista_verbos)
```

```
df_verbos.columns = ['verbos', 'classificacao_nilc']
df_verbos['lema_nilc'] = lista_lemas_verbos
print('Lematizando verbos com Spacy')
df_verbos['lema_spacy'] = df_verbos['verbos'].apply(lambda x: ' '.join([token.lemma_ for token in nlp(x)]))
print('Inserindo raiz morfológica de verbos com NLTK')
df_verbos['raiz_nltk'] = df_verbos['verbos'].apply(lambda x: stemmer.stem(x))
lista_class = df_verbos['classificacao_nilc'].apply(lambda x: str(x)).tolist()
lista_class = '\n'.join(lista_class)
print('Inserindo classificação verbal baseado no Nilc')
lista_class = re.sub(r"'W'", "'Infinitivo'", lista_class)
lista_class = re.sub("'G'", "'Gerúndio'", lista_class)
lista_class = re.sub(r"'K'", "'Particípio'", lista_class)
lista_class = re.sub(r"'I'", "'Pretérito Imperfeito do Indicativo'", lista_class)
lista_class= re.sub(r"'J'", "'Pretérito Perfeito do Indicativo'", lista_class)
lista_class = re.sub(r"'F'", "'Futuro do Presente do Indicativo'", lista_class)
lista_class = re.sub(r"'Q'", "'Pretérito mais que Perfeito do Indicativo'", lista_class)
lista_class = re.sub(r"'S'", "'Presente do Subjuntivo'", lista_class)
lista_class = re.sub(r"'T'", "'Imperfeito do Subjuntivo'", lista_class)
lista_class = re.sub(r"'U'", "'Futuro do Subjuntivo'", lista_class)
lista_class = re.sub(r"'Y'", "'Imperativo'", lista_class)
lista_class = re.sub(r"'C'", "'Futuro do Pretérito'", lista_class)
lista_class = re.sub(r"'P'", "'Presente do Indicativo'", lista_class)
df_verbos['classificacao_verbal'] = [x for x in lista_class.splitlines()]
df_verbos = verbos.merge(df_verbos, on = 'verbos')
df_verbos['syl_size'] = df_verbos['verbos'].apply(lambda x: len(re.findall(r".?uai.?|.?uão.?|.?ai.?|.?ói.?|.?ua.?|.?uo.?|.?io.?|.?ió.?|.?éi.?|.?ei.?|.?ie.?|.?ói.?|.?oi.?|.?au.?|.?ou.?|.?éu.?|.?ui.?|.?a.?|.?á.?|.?â.?|.?ã.?|.?é.?|.?ê.?|.?e.?|.?o.?|.?ô.?|.?õ.?|.?ó.?|.?ò.?|.?i.?|.?í.?", str(x), flags = re.IGNORECASE)))
df_verbos['classificacao_clean'] = df_verbos['classificacao_verbal'].apply(lambda x: re.sub(r'.\d+\w.', '', x))
df_verbos['classificacao_clean'] = df_verbos['classificacao_clean'].apply(lambda x: re.sub(r"\s,\s(?=')", '', x))
df_verbos['classificacao_clean'] = df_verbos['classificacao_clean'].apply(lambda x: re.sub(r",(?=\s$)", '', x))
df_verbos['infinitivo'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Infinitivo', x)))
df_verbos['gerundio'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Gerúndio', x)))
df_verbos['participio'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Particípio', x)))
df_verbos['pret_perf_ind'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito Perfeito do Indicativo', x)))
```

```
df_verbos['pret_imp_ind'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito Imperfeito do Indicativo', x)))
df_verbos['futuro_pret_ind'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do Pretérito', x)))
df_verbos['pres_indicativo'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Presente do Indicativo', x)))
df_verbos['futuro_pres_ind'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do Presente do Indicativo', x)))
df_verbos['pret_mais_perf_ind'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Pretérito mais que Perfeito do Indicativo', x)))
df_verbos['pres_subj'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Presente do Subjuntivo', x)))
df_verbos['imp_subj'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Imperfeito do Subjuntivo', x)))
df_verbos['imperativo'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Imperativo', x)))
df_verbos['futuro_subj'] = df_verbos['classificacao_clean'].apply(lambda x: len(re.findall(r'Futuro do Subjuntivo', x)))
df_verbos = df_verbos[df_verbos['lema_nilc'].str.endswith('r')]
df_verbos.drop('classificacao_clean', axis = 1, inplace = True)
df_verbos.to_csv(f'{file_plot}_verbos_dist.csv')
print('Plotando gráfico das mais mais frequentes - sem stopwords')
stops_spacy = set(STOP_WORDS)
stop_words_coral = {"es", 'ah', 'hum', 'mim', 'comigo', 'ô', 'se', 'ti', 'outro', "ea", "pa'", "ni", "cum", "nimim", "p'", "pa", "di", "o'", "'", 'nan', 'pode', 'tenho'}
stop_words = stops_spacy.union(stop_words_coral)
lista_palavras = df[pd.notnull(df['normalized_utterances'])]
lista_palavras = lista_palavras['normalized_utterances'].tolist()
lista_palavras = '\n'.join(lista_palavras)
lista_palavras = FreqDist([x for x in lista_palavras.split() if x not in stop_words and x not in punctuation])
lista_palavras_df = pd.DataFrame([lista_palavras]).transpose()
lista_palavras_df.reset_index(inplace=True)
lista_palavras_df.columns = ['Palavras', 'Frequência']
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = lista_palavras_df.sort_values(by='Frequência', ascending = False)[:30], x= 'Palavras', y = 'Frequência', marker = 's',  lw='3', color = 'orange' )
b.set_title(f'Palavras mais frequentes em {file_plot}- sem stopwords', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
```

```
b.set_xlabel('Palavras',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig(f'{file_plot}_palavras_mais_frequentes_sem_stopwords.png', dpi=300)
print('Plotando gráfico das palavras mais frequentes - com stopwords')
stops_spacy = set(STOP_WORDS)
stop_words_coral = {"es", 'ah', 'hum', 'mim', 'comigo', 'ô', 'se', 'ti', 'outro', "ea", "pa'", "ni", "cum", "nimim", "p'", "pa", "di", "o'", "'", 'nan', 'pode', 'tenho'}
stop_words = stops_spacy.union(stop_words_coral)
lista_palavras = df[pd.notnull(df['normalized_utterances'])]
lista_palavras = lista_palavras['normalized_utterances'].tolist()
lista_palavras = '\n'.join(lista_palavras)
lista_palavras = FreqDist([x for x in lista_palavras.split() if x not in punctuation])
lista_palavras_df = pd.DataFrame([lista_palavras]).transpose()
lista_palavras_df.reset_index(inplace=True)
lista_palavras_df.columns = ['Palavras', 'Frequência']
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = lista_palavras_df.sort_values(by='Frequência', ascending = False)[:30], x= 'Palavras', y = 'Frequência', marker = 's',  lw='3', color = 'green' )
b.set_title(f'Palavras mais frequentes em {file_plot}- com stopwords', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Palavras',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig('palavras_mais_frequentes_sem_stopwords.png', dpi=300)
print('Plotando gráfico de distribuição dos lemas verbais mais frequentes')
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
b = sns.lineplot(data = df_verbos.sort_values(by='frequencia', ascending = False)[:30], x= 'lema_nilc', ci = False,  y = 'frequencia', marker = 's',  lw='3', color = 'orange')
b.set_title(f'Lemas verbais mais frequentes em {file_plot}', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Lemas verbais',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig('verbos_lemas_lineplot.png', dpi=300)
print('Plotando gráfico de distribuição das formas verbais mais frequentes')
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (9, 5))
plt.xticks(rotation=90)
```

```
b = sns.lineplot(data = df_verbos.sort_values(by='frequencia', ascending = False)[:30], x= 'verbos', ci = False,  y = 'frequencia', marker = 's',  lw='3', color = 'orange')
b.set_title(f'Formas verbais mais frequentes em {file_plot}', fontsize = 18)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Verbos',fontsize=15)
b.tick_params(labelsize=16)
plt.savefig('verbos_lineplot.png', dpi=300)
print('Plotando gráfico com tempos e modos verbais mais frequentes')
tempo_verbos = pd.DataFrame(df_verbos.loc[:, 'infinitivo': ].sum())
tempo_verbos.reset_index(inplace=True)
tempo_verbos.columns = ['Classificação_verbal', 'Frequência']
sns.set_style('whitegrid')
plt.figure(dpi = 300, figsize = (6, 4))
plt.xticks(rotation=90)
b = sns.lineplot(data = tempo_verbos.sort_values(by = 'Frequência', ascending = False), x= 'Classificação_verbal', y = 'Frequência', marker = 's',  lw='3', color = 'orange' )
b.set_title(f'Classificação verbal mais frequente em {file_plot}', fontsize = 16)
b.set_ylabel("Frequência", fontsize = 15)
b.set_xlabel('Classificação verbal',fontsize=14)
b.tick_params(labelsize=15)
plt.savefig('classificacao_verbal_lineplot.png', dpi=300)
#plotando Word Cloud
data = df[['audio', 'normalized_utterances']]
data = data[pd.notnull(data['normalized_utterances'])]
data_list = '\n'.join(data['normalized_utterances'].tolist())
data_words = ' '.join([x for x in data_list.split() if x not in stop_words and x not in punctuation])
if os.path.isfile('figura.png'):
  print('Plotando word cloud com a figura que você escolheu')
  figure = np.array(Image.open('figura.png'))
  plt.figure(figsize=(8,6), dpi = 1000)
  wordcloud = WordCloud(mask = figure, collocations=True, background_color= 'black', max_font_size=90, width=500, height=450, colormap = 'BuPu').generate(data_words)
  plt.imshow(wordcloud)
  plt.axis("off")
  plt.show()
  wordcloud.to_file(f'word_cloud_black.png')
  plt.figure(figsize=(8,6), dpi = 1000)
  wordcloud = WordCloud(mask = figure, collocations=True, background_color= 'white', max_font_size=90, width=500, height=450, colormap = 'inferno').generate(data_words)
  plt.imshow(wordcloud)
```

```
  plt.axis("off")
  plt.show()
  wordcloud.to_file(f'word_cloud_white.png')
else:
  print('Plotando word cloud')
  plt.figure(figsize=(8,6), dpi = 1000)
  wordcloud = WordCloud(collocations=True, background_color= 'black', max_font_size=90, width=500,
height=450).generate(data_words)
  plt.imshow(wordcloud)
  plt.axis("off")
  plt.show()
  wordcloud.to_file(f'word_cloud_black.png')
  plt.figure(figsize=(8,6), dpi = 1000)
  wordcloud = WordCloud(collocations=True, background_color= 'white', max_font_size=90, width=500,
height=450).generate(data_words)
  plt.imshow(wordcloud)
  plt.axis("off")
  plt.show()
  wordcloud.to_file(f'word_cloud_white.png')
```

Anexo C - Programa 3: Escritor de cabeçalho

Link para o programa: https://colab.research.google.com/drive/1qzuUG_eddIU6l-1Z7EKVmFdu0Svok1H9?usp=sharing
Link para atualizações:
https://github.com/carlosjuniorcosta1/escritor_cabecalho_coral_esq_coral_brasil

Finalidade: Localiza as formas aferéticas e convencionalizadas da transcrição fornecida e as escreve no local apropriado dos metadados dos C-ORAIS. Além disso, esse programa já fornece a quantidade de palavras da transcrição ainda sem ter sido alinhada, bem como a quantidade de palavras de algum participante específico. Na Figuras 80 e 81, estão as partes que o programa preenche no cabeçalho, que são quantidade de palavras totais; quantidade de palavras por participante escolhido; formas aferéticas e convencionalizadas.
Arquivos de entrada:
Transcrição do C-ORAL-ESQ ou C-ORAL-BRASIL, em txt (utf-8), ou transcrição que o linguista esteja fazendo – e que seja baseada nos critérios da Teoria da Língua em Ato (cf. MELLO et al, 2012, pp.125-145) – e um modelo de cabeçalho vazio.
Arquivo de saída:
Novo cabeçalho com os campos já referidos preenchidos.

Figura 80 – Exemplo de parte do cabeçalho vazio que é preenchido pelo programa.

```
@Words:
@Patient_words:
@Acoustic_quality:
@Transcriber:
@Revisor:
@Comments:
1) Formas aferéticas:
2) Formas convencionalizadas:
```

Figura 81 – Campos do cabeçalho preenchido pelo programa.

```
@Words: 3178
@Patient_words: 2115
Acoustic_quality:
@Transcriber:
@Revisor:
@Comments:
Apheretic forms: cabou (acabou), tendi (entendi), trapalhado
(atrapalhado);
Conventionalized forms: aques (aqueles), cê (você), cês
(vocês), dum (de um), duma (de uma), ea (ela), eas (elas),
entramo (entramos), es (eles), fumo (fomos), num (em um), numa
(em uma), nũ (não), ocê (você), o' (olha), pa (para), po (para
o), pouquim (pouquinho), pra (para), pro (para o), tava
(estava), tem (tenho), teve (esteve), tivesse (estivesse),
trago (trazido), tá (está), tão (estão), tô (estou), vim (vir),
vão (vamos), xá (deixa);
```

Fonte: Elaborado pelo autor.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 22 21:33:29 2021
@author: José Carlos Costa
"""
import pandas as pd
import re
def limpa_coral(texto):
    import re
    texto = re.sub(r'(\*\w{3}\:)?(\[\d+\])?|\[?\/\d+\]?|\+|/|/{2}|=?i?\s?\-?\w{3}_?r?s?n?=\s?\$?', '', texto)
    texto = texto.replace('hhh', '').replace('yyyy', '')\
    .replace('yyy', '').replace('xxx', '').replace('<', '')\
    .replace('>','').replace('?', '').replace('=', '').replace('[', '').replace(']', '')
    texto = re.sub(r'&\w+', '', texto)
    texto = re.sub(r'\s+', ' ', texto)
    return texto
regex_headers_tuples = r'(\w+|\w+'\w+'|\w+'\w+|\w+')\s\((\w+.*?)\)'
regex_excep= r"([A-Za-z]+')([A-Za-z]+)" #para separar \w+'\w+
regex_excep2 = r'(\w+')\s(\w+')' #para juntar palavras c dois apóstrofos
regex_excep3 = r'(\w+')(\w+')' #para separar as palavras c dois apóstrofos
regex_apostr = r"(\w+'(?!\n))\s(\w+(?!\n))" #para juntar \w' com \w+
regex_remove_participants =
r"(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n.+\n\s+.+\n\s+.+(?!@))|(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n.+\n\
s+.+(?!@).+)|(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n(?!@).+)|(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+(?!@).+)|(
@.+,?(?=.\(.+\))\s+\(.+\)\n.+s+(?!@).+)"
file_1 = str(input('Type the filename you want to extract from: '))
```

```
file_2 = str(input("Type the header's filename - txt - UTF-8: "))
file_3 = str(input("Patient/participant's name you want to count words - example: NAM "))
with open(file_1, 'r+', encoding='utf-8') as source:
    text_tr = source.read()
    text_tr = text_tr.replace("'", "’")
    text_tr_df = pd.DataFrame(text_tr.splitlines())
    text_tr_df = pd.DataFrame([text_tr_df[0].apply(limpa_coral).str.strip()]).transpose()
    text_tr = '\n'.join(text_tr_df[0].tolist())
    text_tr = re.sub(regex_apostr, r"\1\2", text_tr)
with open(file_1, 'r+', encoding='utf-8') as source:
    palavras_totais = source.read()
    palavras_totais = palavras_totais.replace("'", "’")
    palavras_totais = [x for x in palavras_totais.splitlines()]
    palavras_paciente = len(limpa_coral(('\n'.join([x for x in palavras_totais if x[1:4] == file_3]))).split())
    palavras_totais = len(limpa_coral(('\n'.join([x for x in palavras_totais ]))).split())
formas_afereticas = """
babacar (embabacar), brigado (obrigado), brigada (obrigada), baixa (abaixa), credita (acredita), creditei (acreditei), creditou (acreditou) , baixar (abaixar), baixei (abaixei), baulado (abaulado), bora (embora), borrecido (aborrecido), brigada (obrigada), brigado (obrigado), caba (acaba), cabar (acabar), cabava (acabava), cabei (acabei), cabou (acabou), celera (acelera), celerando (acelerando), certar (acertar), chei (achei), cho (acho), contece (acontece), contecer (acontecer), conteceu (aconteceu), cordava (acordava), creditei (acreditei), dianta (adianta), doro (adoro), dotada (adotada), fessora (professora), final (afinal), fundar (afundar), garrado (agarrados), garrou (agarrou), gora (agora), gual (igual), gualzim (igualzinho), guenta (aguenta), guentando (aguentando), guentar (aguentar), guento (aguento), guentou (aguentou), inda (ainda), judar (ajudar), lambique (alambique), laranjado (alaranjado), lisou (alisou), magina (imagina), mamentar (amamentar), manhã (amanhã), marelo (amarelo), marrava (amarrava), migão (amigão aumentativo), mor (amor), ném (neném), panhava (apanhava), parece (aparece), pareceu (apareceu), partamento (apartamento), pelido (apelido),  perta (aperta), pertar (apertar), pertei (apertei), pesar (apesar), pinhada (apinhada), pois (depois), posa (raposa), proveita (aproveita), proveitei (aproveitei), proveitou (aproveitou),  proveitando (aproveitando), proveitei (aproveitei), purra (empurra), qui (daqui), rancaram (arrancaram), rancava (arrancava), rancou (arrancou), ranjar (arranjar), ranjasse (arranjasse), ranjou (arranjou), rebentando (arrebentando), rebentar (arrebentar), regaço (arregaços), rorosa (horrorosa), roz (arroz), rumaram (arrumaram), sobiando (assobiando),té (até), té (até), teirinho (inteirinho), teja (esteja), tendeu (entendeu), tendi (entendi), testino (intestino), tradinha (entradinha), trapalha (atrapalha), trapalhado (atrapalhado), trapalhou (atrapalhou), travessa (atravessa), travessadinho (atravessadinho), trevida (atrevida), trevido (atrevido), trevidão (atrevidão), vó (avó), vô (avô)"""
formas_afereticas = formas_afereticas.replace("'", "’").replace(";", ",")
formas_afereticas = re.sub(regex_apostr, r"\1\2", formas_afereticas)
tuplas_afereticas = re.findall(regex_headers_tuples, formas_afereticas)
```

```
dict_afereticas1 = dict(tuplas_afereticas)
lista_compartilhada_af1 = []
for palavra_tr in text_tr.split():
    for chave, valor in dict_afereticas1.items():
        if palavra_tr == chave:
            lista_compartilhada_af1.append((chave, valor))
text_tr = re.sub(regex_excep, r"\1 \2", text_tr)
text_tr = re.sub(regex_excep2, r"\1\2", text_tr)
formas_afereticas = re.sub(regex_excep, r"\1 \2", formas_afereticas)
formas_afereticas = re.sub(regex_excep2, r"\1\2", formas_afereticas)
tuplas_afereticas2 = re.findall(regex_headers_tuples, formas_afereticas)
dict_afereticas2 = dict(tuplas_afereticas2)
lista_compartilhada_af2 = []
for palavra_tr in text_tr.split():
    for chave, valor in dict_afereticas2.items():
        if palavra_tr == chave:
            lista_compartilhada_af2.append((chave, valor))
formas_conv = """
```

a’ (olha), acabamo (acabamos), achamo (achamos), agradecemo (agradecemos), a’ lá (olha), a’ (olha), a’ (olha), aprendemo (aprendemos), arrumamo (arrumamos), assinávamo (assinávamos), atravessamo (atravessamos), avi (vi), avinha (vinha), bebemo (bebemos), beijamo (beijamos), botemo (botamos), chegamo (chegamos), cheguemo (chegamos), choramo (choramos), colocamo (colocamos), começamo (começamos), comemo (comemos), comemoramo (comemoramos), compramo (compramos), conhecemo (conhecemos), conseguimo (conseguimos), contamo (contamos), conversamo (conversamos), corremo (corremos), cortamo (cortamos), deixamo (deixamos), descansamo (descansamos), descemo (descemos), devemo (devemos), empurramo (empurramos), encontramo (encontramos), entramo (entramos), envem (vem), envinha (vinha), escolhemo (escolhemos), esquecemo (esquecemos), estamo (estamos), estudemo (estudamos), evem (vem), falamo (falamos), fazido (feito), ficamo (ficamos), fize (fiz), fizemo (fizemos), fomo (fomos), for (formos), fraga (flagra), fragando (flagrando), frago (flagro), fumo (fomos), ganhamo (ganhamos), levamo (levamos), levantamo (levantamos), levantemo (levantamos), mandamo (mandamos), manti (mantive), o’ (olha), o’(olha), paramo (paramos), passamo (passamos), pedimo (pedimos), peguemo (pegamos), perdemo (perdemos), pinchando (pichando), pintemo (pintemos), podemo (podemos), precisamo (precisamos), pusemo (pusemos), resolvemo (resolvemos), saímo (saímos), seje (seja), sentamo (sentamos), sentemo (sentamos), separamo (separamos), somo (somos), sufro (sofro), temo (temos), tiramo (tiramos), tivemo (tivemos), tomamo (tomamos), trabalhamo (trabalhamos), trago (trazido), vesse (visse), viemo (viemos), vimo (vimos), tó (toma), cê (você), cês (vocês), e’ (ele), ea (ela), eas (elas), es (eles), ocê (você), ocês (vocês), aque’ (aquele), aquea (aquela), aqueas (aquelas), aques (aqueles), ca (com a), co (com o), cos (com os), cum (com um), cuma (com uma), d’ (de) d’(de), d’(de), d’(de), dum (de um), duma (de uma), dumas (de umas), duns (de uns), n’ deerreí (na DRI), ni (em), n’

(onde), num (em um), numa (em uma), numas (em umas), pa (para), pas (para as), p'(para), p'(para), p'(para), p'(para), p'(para), p'(para), p'(para), po (para o), p'(para), pos (para os), p'(para), pra (para), pr'(para), pras (para as), pro (para o), pr'(para), pros (para os), prum (para um), pruma (para uma), pruns (para uns), p'(para), p' (para), p'(para), pum (para um), puma (para uma), c' aqueas (com aquelas), c'(com), c' cê (com você), c' e' (com ele), c' (com), c'(com essas), c'(com), c' ocê (com você), c' ocês (com vocês), daque' (daquele), daquea (daquela), daqueas (daquelas), daques (daqueles), d' cê (de você), de' (dele), dea (dela), d'(de), d'(de), d'(de), des (deles), d' es (de eles), d'(de), d' ocê (de você), d' ocês (de vocês), naque' (naquele), naquea (naquela), naques (naqueles), ne' (nele), n' ocê (em você), n' ocês (em vocês), p' aque' (para aquele), p'(para), p' cê (para você), p' cês (para vocês), p' e' (para ele), p'(para), p' (para), p'(para), p' es (para eles), p' esse (para esse), p' mim (para mim), p' ocê (para você), p' ocês (para vocês), pr'(para), pr'(para), pr' ea (para ela), pr'(para), pr'(para), pr' ocê (para você), pr' ocês (para vocês), p' sio' (para a senhora), p' siora (para a senhora), p'(para), armoçar (almoçar), artinho (altinho), arto (alto), arto (alto), comprica (complica), compricar (complicar), cravícula (clavícula), escardada (escaldada), prano (plano), pranta (planta), pray (play), prissado (plissado), probremas (problemas), sortando (soltando), sortar (soltar), sortei (soltei), sorto (solto), sortou (soltou), vorta (volta), vortar (voltar), vortava (voltava), vorto (volto), n'é (não é), n'(não), nũ (não), canarim (canarinho), espim (espinho), n' é (não é), padrim (padrinho), passarim (passarinho), porco-espim (porco-espinho), sozim (sozinho), almoçozim (almoçozinho), amarelim (amarelinho), azulzim (azulzinho), bebezim (bebezinho), bichim (bichinho), bocadim (bocadinho), bonitim (bonitinho), cachorrim (cachorrinho), cantim (cantinho), capoeirim (capoeirinhas), carrim (carrinho), cedezinho (CD), certim (certinho), certins (certinhos), Chapeuzim Vermelho (Chapeuzinho Vermelho), chazim (chazinho), controladim (controladinha), desfiadim (desfiadinho), direitim (direitinho), direitim (direitinho), esquisitim (esquisitinho), fechadim (fechadinho), filhotim (filhotinho), formulariozim (formulariozinho), fundim (fundinho), Geraldim (Geraldinho), golezim (golezinho), igualzim (igualzinho), instantim (instantinho), jeitim (jeitinho), Joãozim (Joãozinho), joguim (joguinho), ladim (ladinho), maciim (maciinho), mansim (mansinho), Marquim (Marquinho), meninim (menininho), morenim (moreninho), murim (murinho), Paulim (Paulinho), pequeninim (pequenininha), pertim (pertinho), negocim (negocinhos), partidim (partidinho), porquim (porquinho), portim (portinha), potim (potinho), pouquim (pouquinho), pozim (pozinho), pretim (pretinho), prontim (prontinho), quadradim (quadradinha), quadradim (quadradinho), queimadim (queimadinho), rapidim (rapidinho), recheadim (recheadinho), rolim (rolinho), tamanim (tamaninho), tampadim (tampadinho), terrenim (terreninho), tiquim (tiquinho), todim (todinho), toquim (toquinho), trancadim (trancadinhos), trenzim (trenzinho), tudim (tudinho), sio' (senhora), sior (senhor), siora (senhora), sô (senhor), mó (muito), po' (pode) ,tá (está) ,tamo (estamos) ,tamos (estamos) ,tão (estão) ,tar (estar) ,taria (estaria) ,tás (estás) ,tava (estava) ,tavam (estavam) ,távamos (estávamos) ,tavas (estavas) ,teja (esteja) ,tem (tenho) ,teve (esteve) ,tive (estive) ,tiver (estiver) ,tiverem (estiverem) ,tivesse (estivesse) ,tô (estou) ,vamo (vamos) ,vão (vamos) ,vim (vir) ,xá (deixa), antiguim (antiguinho), banhozim (banhozinho), branquim (branquinho), certim (certinho), devagarzim (devagarzinho), direitim (direitinho), direitim (direitinho), gostosim (gostosinho), limãozim (limãozinho), minutim (minutinho), pertim (pertinho), pulim (pulinho), pouquim (pouquinho), rapidim (rapidinho), recibim (recibinho), verdim (verdinho), xixizim (xixizinho), zerim (zerinho)

```
"""
text_tr = re.sub(regex_excep3, r'\1 \2', text_tr )
text_tr = re.sub(regex_apostr, r"\1\2", text_tr)
#onomatopeias = """
#tanana, bla bla bla
formas_conv = formas_conv.replace("'", "’").replace(";", ",")
formas_conv = re.sub(regex_apostr, r"\1\2", formas_conv)
tuplas_conv = re.findall(regex_headers_tuples, formas_conv)
dict_conv1 = dict(tuplas_conv)
lista_compartilhada_conv1 = []
for palavra_tr in text_tr.split():
    for chave, valor in dict_conv1.items():
        if palavra_tr == chave:
            lista_compartilhada_conv1.append((chave, valor))
text_tr = re.sub(regex_excep, r"\1 \2", text_tr)
text_tr = re.sub(regex_excep3, r'\1\2', text_tr)
text_tr = re.sub(regex_apostr, r'\1\2', text_tr)
formas_conv = re.sub(regex_excep, r"\1 \2", formas_conv)
formas_conv = re.sub(regex_excep2, r'\1\2', formas_conv)
formas_conv = re.sub(regex_apostr, r'\1\2', formas_conv)
tuplas_conv2 = re.findall(regex_headers_tuples, formas_conv)
dict_conv2 = dict(tuplas_conv2)
lista_compartilhada_conv2 = []
for palavra_tr in text_tr.split():
    for chave, valor in dict_conv2.items():
        if palavra_tr == chave:
            lista_compartilhada_conv2.append((chave, valor))
with open(file_2, 'r+', encoding='utf-8' ) as source:
    header = source.read()
    text_h = header.replace("'", "’")
    text_h = re.sub(r'\w+\s(?=\(vocativo\)).\w+..', '', text_h)
    text_h = re.sub(r'\w+\s(?=\(interjeição\)).\w+..', '', text_h)
    text_h = re.sub(regex_remove_participants, '', text_h)
    text_h = re.sub(r'(@.+(?!\n.+@))', '', text_h)
    text_tr = re.sub(regex_excep, r'\1 \2', text_tr)
    text_h = re.sub(regex_apostr, r"\1\2", text_h)
    text_tr = re.sub(regex_apostr, r"\1\2", text_tr)
    tuplas_h1 = re.findall(regex_headers_tuples, text_h)
    dict_h1 = dict(tuplas_h1)
    lista_compartilhada_h1 = []
```

```
    for palavra_tr in text_tr.split():
        for chave, valor in dict_h1.items():
            if palavra_tr == chave:
                lista_compartilhada_h1.append((chave, valor))
    text_tr = re.sub(regex_excep3, r'\1 \2', text_tr) #\w+’\w+’ > \w+’\s\w+’
    text_tr = re.sub(regex_excep, r"\1 \2", text_tr) #\w+’\w+ > \w’\s\w+ #espaçamento original aqui
    text_h = re.sub(regex_excep3, r'\1 \2', text_h)
    text_h = re.sub(regex_excep, r"\1 \2", text_h)
    tuplas_h2 = re.findall(regex_headers_tuples, text_h)
    dict_h2 = dict(tuplas_h2)
    lista_compartilhada_h2 = []
    for palavra_tr in text_tr.split():
        for chave, valor in dict_h2.items():
            if palavra_tr == chave:
                lista_compartilhada_h2.append((chave, valor))
    for x in lista_compartilhada_af2:
        lista_compartilhada_af1.append(x)
    for x in lista_compartilhada_conv2:
        lista_compartilhada_conv1.append(x)
    for x in lista_compartilhada_h2:
        lista_compartilhada_h1.append(x)
text_h                                                                                                  =
re.sub(r'Formas\s+?aferéticas:.+|Formas\s+?convencionalizadas:.+|Apheretic\s+?forms:.+|Apheretic\s
+?Forms:.+|Conventionalized\s+?forms:.+|Standardized\s+?forms:.+|Standardized\s+?Forms:.+',      '',
text_h)
text_h_restante = re.sub(r'\d\)\s', '', text_h)
text_h_restante = '\n'.join(re.findall(r'.+', text_h_restante))
dict_afereticas_total = dict(lista_compartilhada_af1)
dict_conv_total = dict(lista_compartilhada_conv1)
dict_h_total = dict(lista_compartilhada_h1)
chaves_conv = set(dict_conv_total.items())
chaves_af = set(dict_afereticas_total.items())
chaves_h = set(dict_h_total.items())
conjunto1 = chaves_af.difference(chaves_h)
conjunto2 = chaves_h.difference(chaves_af)
conjunto3= chaves_af.intersection(chaves_h)
conjunto_f = chaves_h - conjunto2
formas_af_totais = dict(conjunto1.union(conjunto_f))
conjunto4 = chaves_conv.difference(chaves_h)
conjunto5 = conjunto2.difference(chaves_conv)
```

```
conjunto6 = chaves_conv.intersection(conjunto2)
formas_conv_totais = dict(conjunto4.union(conjunto5).union(conjunto6))
with open(file_2, 'r+', encoding='utf-8' ) as source:
    header = source.read()
    header = header.replace("""", "'")
    header_df = pd.DataFrame()
    header_df['Title'] = ["@Title:" + ' '.join(re.findall(r'(?<=@Title:).+', header))]
    header_df['File'] = "@File:" + ' '.join(re.findall(r'(?<=@File:).+', header))
    header_df['Participants'] =
''.join(str(re.findall(r'(?<=Participants:)(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n.+\n\s+.+\n\s+.+(?!@))|(@.+,
?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n.+\n\s+.+(?!@).+)|(@.+,?(?=.\(.+\))\s+\(.+\)\n.+\n\s+.+\n(?!@).+)|(@.+,?(
?=.\(.+\))\s+\(.+\)\n.+\n\s+(?!@).+)|(@.+,?(?=.\(.+\))\s+\(.+\)\n.+s+(?!@).+)', header)))
    header_tratamento = '\n'.join(header_df['Participants'].tolist())
    header_tratamento = re.sub("\",\s+|\[\(|,\s\"\)\]|'", '', header_tratamento)
    header_tratamento = re.sub(r'\\n', '\n', header_tratamento)
    header_tratamento = re.sub(r'\\t', '\t', header_tratamento)
    header_df['Participants'] = header_tratamento
    header_df['Participants'] = header_df['Participants'].apply(lambda x: re.sub(r'\)\](?=$)', '', x))
    header_df['Date'] = "@Date:" + ' '.join(re.findall(r'(?<=@Date:).+', header))
    header_df['Place'] = "@Place:" + ' '.join(re.findall(r'(?<=@Place:).+', header))
    header_df['Situation'] = "@Situation:" + ' '.join(re.findall(r'(?<=@Situation:).+', header))
    header_df['Topic'] = "@Topic:" + ' '.join(re.findall(r'(?<=@Topic:).+', header))
    header_df['Source'] = "@Source:" + ' '.join(re.findall(r'(?<=@Source:).+', header))
    header_df['Class'] = "@Class:" + ' '.join(re.findall(r'(?<=@Class:).+', header))
    header_df['Length'] = "@Length:" + ' '.join(re.findall(r'(?<=@Length:).+', header))
    header_df['Words'] = "@Words:" + ' ' + str(palavras_totais)
    header_df['Patient_words'] = "@Patient_words:" + ' ' + str(palavras_paciente)
    header_df['Acoustic_quality'] = "Acoustic_quality: "
    header_df['Transcriber'] = "@Transcriber:" + ' '.join(re.findall(r'(?<=@Transcriber:).+', header))
    header_df['Revisor'] = "@Revisor:" + ' '.join(re.findall(r'(?<=@Revisor:).+', header))
    header_df['Comments'] = "@Comments:"
    header_df.transpose()
    lista_af_final = []
    for k, v in formas_af_totais.items():
        lista_af_final.append(f'{k} ({v}),')
    header_df['Apheretic_forms'] = "Apheretic forms: " + ' '.join(sorted(lista_af_final))
    lista_conv_final = []
    for k, v in formas_conv_totais.items():
        lista_conv_final.append(f'{k} ({v}),')
    header_df['Conventionalized_forms'] = "Conventionalized forms: " + ' '.join(sorted(lista_conv_final))
```

```
    header_df['Conventionalized_forms'] = header_df['Conventionalized_forms'].apply(lambda x: re.sub(r',(?=$)', ';', x))
    header_df['Conventionalized_forms'] = header_df['Conventionalized_forms'].apply(lambda x: re.sub(regex_excep, r'\1 \2', x))
    header_df['Apheretic_forms'] = header_df['Apheretic_forms'].apply(lambda x: re.sub(r',(?=$)', ';', x))
    header_df_derretido = header_df.melt()
    texto_rest_df = pd.DataFrame([text_h_restante.splitlines()]).transpose()
    texto_rest_df.reset_index(inplace = True)
    texto_rest_df.columns = ['variable', 'value']
    texto_rest_df['variable'] = texto_rest_df['variable'].astype('str')
    header_df_derretido = pd.concat([header_df_derretido, texto_rest_df], ignore_index = True)
    header_txt = '\n'.join(header_df_derretido['value'].tolist())
    with open(f"{file_2[:-4]}_novo.txt", 'w+', encoding = 'utf-8') as writer:
        file = writer.write(header_txt)
        print('By: José Carlos Costa \n Let me know if I can help you with something :) \n whatsapp: +55 31 98924 1307 \n email: carlosjuniorcosta1@gmail.com')
```

Anexo D – Programa 4 – extrai pausas silenciosas de TextGrid geradas a partir de Lennes (2002, 2006)

Link para o programa:
https://colab.research.google.com/drive/1GNJCs3ASUkNVvVG9fPCtnCirXNxY6jHb?usp=sharing

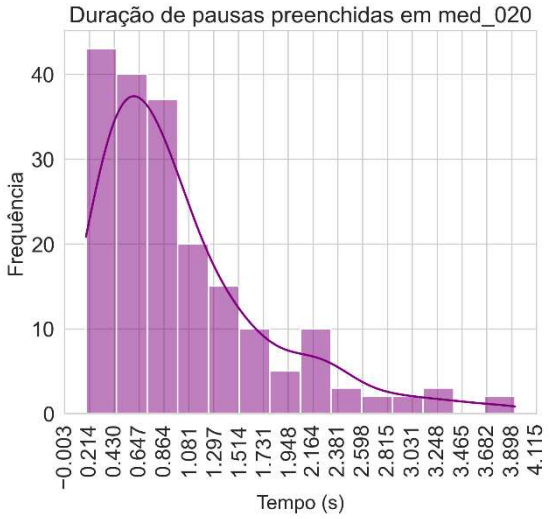
Link para atualizações:
https://github.com/carlosjuniorcosta1/silent_pauses_from_text_grid/edit/main/README.md

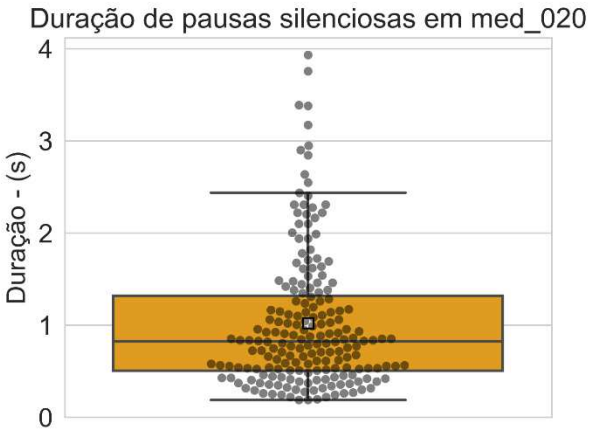
Finalidade: Este programa extrai as pausas silenciosas geradas a partir de Lennes (2002, 2006) – marcadas com “xxx” no Praat - e gera visualizações desse fenômeno em um histograma, boxplot e gráfico de enxame.

Instruções: Suba o arquivo TextGrid gerado pelo código de Lennes e clique no executar. É necessário que as pausas estejam marcadas com “xxx” (opção configurável na janela do Praat). É possível estabelecer limite mínimo e máximo para as pausas silenciosas, em segundos.
Tutotrial em vídeo: https://www.loom.com/share/e11cfb15150143108ec2692c929f1669

Arquivos de saída:
Dataframe com o valor da pausa silenciosa calculado.
Visualizações em histograma e boxplot com a duração das pausas silenciosas.
Instruções: https://www.loom.com/share/caff81304caa4aaf816ee12711910ec4

Exemplos de visualizações de saída:





"""

Created on Thu Dec 16 18:58:36 2021
@author: José Carlos Costa
email: carlosjuniorcosta1@gmail.com
"""

```
import pandas as pd
import re
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
file = input(str('Digite o nome - com extensão TextGrid- do arquivo gerado pelo script de Mietta Lennes \n Lembre-se de que as pausas devem estar marcadas com "xxx"'))
while True:
    limite_min = input('Digite um limite mínimo,em segundos, para ser considerado pausa silenciosa no gráfico \n ou aperte enter para ignorar: ')
    if len(limite_min) == 0:
        pass
    if len(limite_min) > 0:
        limite_min = float(limite_min)
    limite_max = input('Digite um limite máximo, em segundos, para ser considerado pausa silenciosa no gráfico \n ou aperte enter para ignorar: ')
    if len(limite_max) == 0:
        continue
    if len(limite_max) > 0:
        limite_max = float(limite_max)
    break
```

```
with open(file, 'r') as source:
    arquivo = source.read()
arquivo_f = ' '.join(re.findall(r'intervals.+\n.\s+xmin...+\n.\s+xmax.+\n.\s+text...\"xxx\"', arquivo))
arquivo_f = re.sub(r'(?<=text).+', '', arquivo_f)
arquivo_f = arquivo_f.split('text')
df = pd.DataFrame(arquivo_f)
df.columns = ['dados']
df['xmin'] = df['dados'].apply(lambda x: ' '.join(re.findall(r'(?<=xmin).+', x)))
df['xmin'] = df['xmin'].apply(lambda x: re.sub(r'=\s', '', x))
df['xmax'] = df['dados'].apply(lambda x: ' '.join(re.findall(r'(?<=xmax).+', x)))
df['xmax'] = df['xmax'].apply(lambda x: re.sub(r'=\s', '', x))
df = df.query('xmin != ""')
df['xmin'] = df['xmin'].astype('float')
df['xmax'] = df['xmax'].astype('float')
df['silent_pause'] = df['xmax'] - df['xmin']
df['silent_pause'] = df['silent_pause'].round(3)
df['audio'] = file[:-9]
if type(limite_min) == float and type(limite_max) == float:
    df_f = df.query('silent_pause > @limite_min & silent_pause < @limite_max')
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(6, 5))
    a = sns.histplot(data= df_f, x = 'silent_pause', kde = True, color = 'purple')
    a.set_title(f'Duração de pausas preenchidas em {df_f["audio"][0]}', fontsize = 16)
    a.set_xlabel("Tempo (s)",fontsize= 14)
    a.set_ylabel("Frequência",fontsize = 15)
    a.tick_params(labelsize=15)
    a.xaxis.set_major_locator(ticker.LinearLocator(20))
    plt.xticks(rotation=90)
    plt.show()
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(5, 4))
    a = sns.boxplot(data = df_f, y = 'silent_pause', showmeans = True, showfliers = False, \
                meanprops={"marker":"s","markerfacecolor":"white",  "markeredgecolor":"black"},  color  =
'orange')
    sns.swarmplot(data = df_f, y = 'silent_pause', color = 'black', alpha = 0.5)
    a.set_title(f'Duração de pausas silenciosas em {df_f["audio"][0]}', fontsize = 16)
    # a.set_xlabel(,fontsize= 14)
    a.set_ylabel("Duração - (s)",fontsize = 15)
    a.tick_params(labelsize=15)
    plt.show()
```

```
if type(limite_min) == float and type(limite_max) == str:
    df_f = df.query('silent_pause > @limite_min')
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(6, 5))
    a = sns.histplot(data= df_f, x = 'silent_pause', kde = True, color = 'purple')
    a.set_title(f'Duração de pausas preenchidas em {df_f["audio"][0]}', fontsize = 16)
    a.set_xlabel("Tempo (s)",fontsize= 14)
    a.set_ylabel("Frequência",fontsize = 15)
    a.tick_params(labelsize=15)
    a.xaxis.set_major_locator(ticker.LinearLocator(20))
    plt.xticks(rotation=90)
    plt.show()
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(5, 4))
    a = sns.boxplot(data = df_f, y = 'silent_pause', showmeans = True, showfliers = False, \
                meanprops={"marker":"s","markerfacecolor":"white",  "markeredgecolor":"black"},  color  =
'orange')
    sns.swarmplot(data = df_f, y = 'silent_pause', color = 'black', alpha = 0.5)
    a.set_title(f'Duração de pausas silenciosas em {df_f["audio"][0]}', fontsize = 16)
    # a.set_xlabel(,fontsize= 14)
    a.set_ylabel("Duração - (s)",fontsize = 15)
    a.tick_params(labelsize=15)
    plt.show()
if type(limite_min) == str and type(limite_max) == float:
    df_f = df.query('silent_pause < @limite_max')
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(6, 5))
    a = sns.histplot(data= df_f, x = 'silent_pause', kde = True, color = 'purple')
    a.set_title(f'Duração de pausas preenchidas em {df_f["audio"][0]}', fontsize = 16)
    a.set_xlabel("Tempo (s)",fontsize= 14)
    a.set_ylabel("Frequência",fontsize = 15)
    a.tick_params(labelsize=15)
    a.xaxis.set_major_locator(ticker.LinearLocator(20))
    plt.xticks(rotation=90)
    plt.show()
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(5, 4))
    a = sns.boxplot(data = df_f, y = 'silent_pause', showmeans = True, showfliers = False, \
                meanprops={"marker":"s","markerfacecolor":"white",  "markeredgecolor":"black"},  color  =
'orange')
```

```
    sns.swarmplot(data = df_f, y = 'silent_pause', color = 'black', alpha = 0.5)
    a.set_title(f'Duração de pausas silenciosas em {df_f["audio"][0]}', fontsize = 16)
    # a.set_xlabel(,fontsize= 14)
    a.set_ylabel("Duração - (s)",fontsize = 15)
    a.tick_params(labelsize=15)
    plt.show()
if type(limite_min) == str and type(limite_max) == str:
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(6, 5))
    a = sns.histplot(data= df, x = 'silent_pause', kde = True, color = 'purple')
    a.set_title(f'Duração de pausas preenchidas em {df["audio"][0]}', fontsize = 16)
    a.set_xlabel("Tempo (s)",fontsize= 14)
    a.set_ylabel("Frequência",fontsize = 15)
    a.tick_params(labelsize=15)
    a.xaxis.set_major_locator(ticker.LinearLocator(20))
    plt.xticks(rotation=90)
    plt.show()
    sns.set_style('whitegrid')
    plt.figure(dpi = 300, figsize=(5, 4))
    a = sns.boxplot(data = df, y = 'silent_pause', showmeans = True, showfliers = False, \
                meanprops={"marker":"s","markerfacecolor":"white",  "markeredgecolor":"black"},  color  =
'orange')
    sns.swarmplot(data = df_f, y = 'silent_pause', color = 'black', alpha = 0.5)
    a.set_title(f'Duração de pausas silenciosas em {df["audio"][0]}', fontsize = 16)
    # a.set_xlabel(,fontsize= 14)
    a.set_ylabel("Duração - (s)",fontsize = 15)
    a.tick_params(labelsize=15)
    plt.show()
try:
    if len(df_f) < 0:
        df.to_csv(f'{df["audio"][0]}.csv')
    if len(df_f) > 0:
        df.to_csv(f'{df["audio"][0]}.csv')
        df_f.to_csv(f'{df_f["audio"][0]}_lim.csv')
except:
    pass
```

Anexo E – Web Scraping na VerboWeb

Finalidade: Extrair informações sobre os verbos na VerboWeb por meio de um robô e incorporar essas informações ao C-ORAL-ESQ e ao C-ORAL-BRASIL.

Instruções de uso: No Spyder ou na IDE ou editor de sua preferência[52], selecionar o código e clicar no executar. Não recomendo o Google Colab para essa finalidade. Digite o nome do arquivo obtido no Programa 2 para os verbos. Aguardar o scraping e a exportação do dataframe.

Tutorial em vídeo: https://www.loom.com/share/8b6277d286be456b85b6af28354ad945

Link para atualizações: https://www.loom.com/share/8b6277d286be456b85b6af28354ad945

Arquivos de saída:

Dataframe com todas as entradas disponíveis atualmente na VerboWeb, com informações sobre classe verbal, aspecto lexical, papel temático, estrutura sintática e transitividade.

Dataframe com as informaçãoes acima incorporado ao lema em questão do C-ORAL-ESQ ou C-ORAL-BRASIL.

```
"""
Created on Tue Nov  2 23:41:22 2021
@author: Usuario
"""
#robô para infiltrar no verboweb
from selenium import webdriver
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
from bs4 import BeautifulSoup
import requests
import re
from webdriver_manager.chrome import ChromeDriverManager
file = input('Digite o nome do arquivo com a distribuição dos verbos \n que você obteve no programa de extração lexical - Programa 2')
driver = webdriver.Chrome(ChromeDriverManager().install())
url = "http://www.letras.ufmg.br/sistemas/verboweb_cliente/lista.php?lg=pt"
driver.get(url)
html_content = requests.get(url).text
soup = BeautifulSoup(html_content, "lxml")
verbos = soup.find_all('td')
verbos = '\n'.join([x.text for x in verbos])
verbos = re.sub(r"\d+\.\s", '', verbos)
verbos = verbos.replace('Ã§', 'ç')
verbos = list(verbos.splitlines())
driver.find_element_by_link_text('Abalar 1').click()
links_verbos = soup.find_all('a')
links_verbos = '\n'.join([str(x) for x in links_verbos])
```

[52]Não recomendo o uso do Google Colab para essa função.

```
links_verbos = '\n'.join(re.findall(r'(?<=\").+(?=\")', links_verbos))
links_verbos = [x for x in links_verbos.splitlines()]
del links_verbos[0]
nova_url = "http://www.letras.ufmg.br/sistemas/verboweb_cliente/"
from time import sleep
lista_verbos = []
aspecto_lista = []
sintagma_lista = []
papel_tematico_lista = []
classe_lista = []
exemplo_verbo_web_lista = []
for y in links_verbos:
    # driver.implicitly_wait(1)
    driver.get(nova_url + y)
    # sleep(1)
    estrutura_sintatica = driver.find_elements_by_css_selector('#opener4')
    if len(estrutura_sintatica) > 0:
        verbo_lema = driver.find_elements_by_tag_name('h3')
        # sleep(1)
        classe = driver.find_elements_by_css_selector('#opener1')
        papel_tematico = driver.find_elements_by_css_selector('#opener5')
        aspecto = driver.find_elements_by_css_selector('#opener7')
        exemplo_verbo_web = driver.find_element_by_tag_name('i').text
        for z in verbo_lema:
            lista_verbos.append(z.text)
        for c in classe:
            classe_lista.append(c.text)
        for w in aspecto:
            aspecto_lista.append(w.text)
        for k in estrutura_sintatica:
            sintagma_lista.append(k.text)
        for b in papel_tematico:
            papel_tematico_lista.append(b.text)
        for e in exemplo_verbo_web.splitlines():
            exemplo_verbo_web_lista.append(e)
    else:
        continue
driver.close()
lista_verbos = [x for x in lista_verbos if len(x)]
df = pd.DataFrame()
df['lema_verbo_web'] = lista_verbos
df['classe_verbo_web'] = classe_lista
df['aspecto_lexical'] = aspecto_lista
df['papel_tematico'] = papel_tematico_lista
df['estrutura_sintatica'] = sintagma_lista
df['exemplo_verbo_web'] = exemplo_verbo_web_lista
df['lema'] = df['lema'].apply(lambda x: re.sub(r'\d+', '', x))
df['lema'] = df['lema'].str.strip()
df_verbos_cesq = pd.read_csv(file)
df_verbos_cesq.columns = ['verbos', 'frequencia', 'classificacao_nilc', 'lema',
      'lema_spacy', 'raiz_nltk', 'classificacao_verbal', 'syl_size',
      'infinitivo', 'gerundio', 'participio', 'pret_perf_ind', 'pret_imp_ind',
      'futuro_pret_ind', 'pres_indicativo', 'futuro_pres_ind',
      'pret_mais_perf_ind', 'pres_subj', 'imp_subj', 'imperativo',
      'futuro_subj']
df_verbos_cesq = df_verbos_cesq.drop_duplicates(subset = 'lema')
df_verbos_merged = df_verbos_cesq.merge(df, on = 'lema')
#exportação
df.to_csv('verbo_web.csv')
df.to_excel('verbo_web.xlsx')
```

```
df_verbos_merged.to_excel('coral_esq_verbo_web.csv')
```

Anexo F – Web scraping na Framenet Brasil

Finalidade: Entrar no Lexicon da Framenet Brasil e extrair os frames que possuem unidades lexicais a eles associadas. A partir dessas e do programa do Anexo G, é possível contabilizar as unidades lexicais dos enunciados do C-ORAL-ESQ ou C-ORAL-BRASIL e relacioná-las às centenas de cenas disponíveis na Framenet – se forem iguais às unidades lexicais do C-ORAL-ESQ ou C-ORAL-BRASIL.

Link para atualizações:
https://github.com/carlosjuniorcosta1/web_scraping_frame_net_brasil

Instruções de uso: Esse programa funciona em duas partes. Primeiramente, é necessário fazer o web scraping na Framenet. Para isso, selecione o código e clique no executar. Utilize a IDE ou editor de texto de sua preferência. Recomendo o Spyder. Essa primeira parte, tal como está, não funciona no Google Colab.

Arquivo de saída:
Dataframe com todas o nome de cada frame que possui unidade lexical associada e está presente no site no momento do scraping, além do nome, unidades lexicais e a definição do frame.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Nov  4 23:37:00 2021
@author: José Carlos Costa
Email: carlosjuniorcosta1@gmail.com
"""
from selenium import webdriver
import pandas as pd
import re
from bs4 import BeautifulSoup
import requests
from webdriver_manager.chrome import ChromeDriverManager
driver = webdriver.Chrome(ChromeDriverManager().install())
url = "http://webtool.framenetbr.ufjf.br/index.php/webtool/report/frame/main"
driver.get(url)
html_content = requests.get(url).text
soup = BeautifulSoup(html_content, "lxml")
frame_title = driver.find_elements_by_class_name('tree-title')
frame_list = []
for x in frame_title:
    frame_list.append(x.text)
frame_list = [x for x in frame_list if x != "Frames"]
```

```
driver.find_elements_by_class_name('tree-node')[6].click()
frame_name_list = []
frame_definition_list = []
lexical_units_list = []
from time import sleep
for x in range(0, len(frame_list)):
    pasta_frames = driver.find_elements_by_class_name('tree-node')[x].click()
    sleep(2)
    if len(driver.find_elements_by_class_name('tableLU')) > 0:
        frame_name = driver.find_elements_by_class_name('frameName')
        frame_definition = driver.find_elements_by_class_name('text')
        lexical_units = driver.find_elements_by_class_name('tableLU')
        for y in frame_name:
            frame_name_list.append(y.text)
        for z in frame_definition:
            frame_definition_list.append(z.text)
        for l in lexical_units:
            lexical_units_list.append(l.text)
    else:
        continue
driver.close()
df = pd.DataFrame()
df['frame_name'] = frame_name_list
df['frame_definition'] = frame_definition_list
df['lexical_units_list'] = lexical_units_list
df.to_csv('framenet_dados.csv')
```

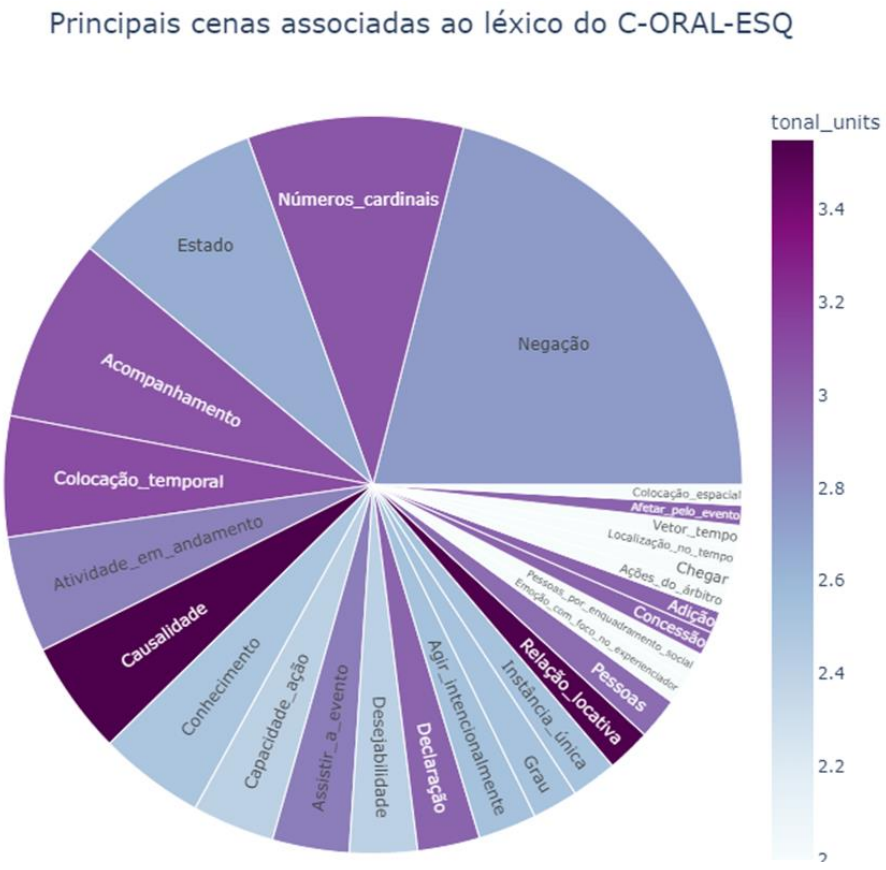
Anexo G – Relacionador de frames das unidades lexicais

Link para o programa:
https://colab.research.google.com/drive/1mKOQsDmcSFU2jxwcA2xFK747YHcp0Kin?usp=sharing

Atualizações: https://github.com/carlosjuniorcosta1/avaliador_de_frames_lexico
Finalidade: Ainda em fase de implementação, mas já funcionando, este programa contabiliza as unidades lexicais a partir dos dados obtidos no scraping da Framenet Brasil e as compara às unidades lexicais, por meio do lema, no C-ORAL-ESQ ou C-ORAL-BRASIL. Trata-se de um procedimento baseado exclusivamente no léxico e que não considera porções mais holísticas, como construções, e tampouco papéis desempenhados nesses frames.Palavras compostas ou construções já cadastradas como unidades lexicais no site da Framenet, na ocasião do scraping, são cadastradas e reconhecidas, porém.

Trata-se de um cálculo que soma todas as unidades lexicais encontradas em cada enunciado e fornece um possível frame predominante. Além disso, esse código gera visualização, como da Figura 82 a seguir

Figura 82- Cenas associadas às unidades lexicais da transcrição utilizada.



Fonte: Elaborado pelo autor.

Instruções de uso: Após ter extraído alguma transcrição por meio do Programa 1, abra o Google Colab, suba o arquivo da transcrição e o obtido no web scraping da Framenet chamado ‘frame_net_dados.csv’. Clique no executar e aguarde. Após a mensagem de erro, reinicie o ambiente de execução pelo menu e , depois, clique no executar novamente.

```
#autor: José Carlos Costa
#email: carlosjuniorcosta1@gmail.com
import pandas as pd
import re
!pip install spacy
!python3 -m spacy download pt
!pip install --upgrade plotly
import spacy
nlp = spacy.load('pt_core_news_sm')
import plotly.graph_objects as go
import plotly.express as px
import numpy as np
import plotly.graph_objects as go
import os
file1 = pd.read_csv(str(input('Filename (C-ORAL-ESQ/BRASIL, csv): ')))
file2 = pd.read_csv('frame_net_dados.csv')
file_plot = ' '.join([x[:-4] for x in os.listdir() if not x.startswith('frame_net_dados') and x.endswith('csv')])
def coral_framenet():
  df = file1.copy()
  df_frame = file1.merge(file2, how = 'left')
  df_frame = df_frame.fillna(' ')
  df_frame['normalized_utterances'] = df_frame['normalized_utterances'].str.lower()
  df_frame['lema'] = df_frame['normalized_utterances'].apply(lambda x: ' '.join([token.lemma_ for token
in nlp(x)]))
      #cria 698 colunas de frames e conta os lexemas dos enunciados
  df_frame["Abundância_distribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcobrir\b|\brevestir\b", str(x))))
  df_frame["Abundância_distribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcobrir\b|\brevestir\b", str(x))))
  df_frame["Abandono"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babandonado\b|\babandonar\b|\babandono\b|\bdeixar\b|\besquecer\b|\besquecido\b|\b
negligenciar\b", str(x))))
  df_frame["Abertura"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baberto\b|\bfechado\b",
str(x))))
  df_frame["Absorção_de_calor"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bassar\b|\bbranquear\b|\bcozinhar\b|\bdourar\b|\bferventar\b|\bferver\b|\bfritar\b|\bgrelh
ar\b|\brefogar\b", str(x))))
  df_frame["Abundância"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babundante\b|\babundar\b|\brico\b", str(x))))
  df_frame["Abundância_distribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcobrir\b|\brevestir\b", str(x))))
  df_frame["Abundar_com"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babarrotado\b|\babundante\b|\badornado\b|\baglomerado\b|\baglomerar\b|\bamanteiga
```

```
do\b|\bamontoado\b|\basfaltado\b|\baspergido\b|\bborrifado\b|\bcheio\b|\bcoberto\b|\bdecorado\b|\bde
sarrumado\b|\bdourado\b|\bdrapeado\b|\bembelezado\b|\bemperrado\b|\bempilhado\b|\bempoeirado\b
|\bencapotado\b|\bencasacado\b|\bencoberto\b|\benfeitado\b|\bengatinhar\b|\benvernizado\b|\bescova
do\b|\besmaltado\b|\bespalhado\b|\bforrado\b|\binjetado\b|\blacado\b|\bladrilhado\b|\blotado\b|\bmanc
hado\b|\bornamentado\b|\bpavimentado\b|\bpendurado\b|\bpintado\b|\bpolvilhado\b|\bpontilhado\b|\bp
opulacional\b|\bpreenchido\b|\bproliferar\b|\brastejante\b|\brebocado\b|\brecheado\b|\bregado\b|\brepl
eto\b|\brespingado\b|\bsalpicado\b|\bsuperlotado\b", str(x))))
  df_frame["Abusar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\babusar\b|\babuso\b",
str(x))))
  df_frame["Acabar_de_descobrir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bchocado\b",
str(x))))
  df_frame["Ação_sucedida"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bem\ssucedido\b|\bem-sucedido\b|\bombar\b|\bdesandar\b", str(x))))
  df_frame["Aceitar_ou_recusar_a_agir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\brecusar\b|\bresistir\b", str(x))))
  df_frame["Acessórios_de_vestuário"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfita\b|\bmáscara\b", str(x))))
  df_frame["Ações_do_árbitro"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapitar\sfim\b|\bapitar\sinício\b|\bapitar\b|\bconceder\b|\bdecidir\b|\bdecisão\b|\bdescla
ssificar\b|\bdesqualificar\b|\bencerrar\b|\bexpulsar\b|\biniciar\b|\binterromper\b|\bmarcar\sfalta\b|\bmar
car\b|\bmostrar\b|\bparalisar\b|\bparar\b|\breiniciar\b|\bsuspender\b|\bterminar\b", str(x))))
  df_frame["Acomodação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacampamento\b|\bacomodação\b|\balbergue\b|\balojamento\b|\bapart-
hotel\b|\bapartamento\b|\bbangalô\b|\bcafofo\b|\bcamping\b|\bcasa\sde\sférias\b|\bcasa\b|\bchácara\b|
\bchalé\b|\bcomplexo\sde\scondomínio\b|\bcomplexo\sresidencial\b|\bestância\b|\bgranja\b|\bhospeda
gem\sdomiciliar\b|\bhospedagem\b|\bhóspede\b|\bhostel\b|\bhotel\sfazenda\b|\bhotel\b|\bhotelaria\b|\b
motel\b|\bpensão\b|\bpousada\b|\brancho\b|\bresort\b|\bsítio\b", str(x))))
  df_frame["Acompanhamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\ssós\b|\bacompanhar\b|\bcom\b|\bcom\b|\bcompanhia\b|\bindividual\b|\bjunto\b|\bso
zinho\b|\bunido\b", str(x))))
  df_frame["Acordar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bacordar\b", str(x))))
  df_frame["Adequação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badequação\b|\badequado\b|\badequar\b|\bambientar\b|\bapropriado\b|\bbom\ssenso\
b|\bbom\b|\bcerto\b|\bclimatização\b|\bclimatizar\b|\bcorreto\b|\binadequação\b|\binadequado\b|\binapr
opriado\b|\bindicado\b|\bprestar\b|\bpróprio\b|\bservir\b", str(x))))
  df_frame["Adição"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacrescentar\b|\badicionar\b|\bmais\b|\bsomar\b", str(x))))
  df_frame["Adjacência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badjacência\b|\badjacente\b|\bcontiguidade\b|\bcontíguo\b|\bestar\sjunto\b|\bjuntar\b|\b
limitante\b|\blimitar\b|\bvizinho\b", str(x))))
  df_frame["Adotar_seleção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badoção\b|\badotar\b|\bassumir\b|\bseguir\b", str(x))))
  df_frame["Adquirir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconseguir\b|\bganhar\b|\bobtido\b|\breconquistar\b|\brecuperar\b", str(x))))
  df_frame["Afetar_pelo_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacontecer\b|\bassolar\b|\batingir\b|\blevar\b|\bsofrer\b|\bver\b", str(x))))
  df_frame["Afirmar_ou_negar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\binegável\b|\bnegar\b|\bnegativo\b", str(x))))
  df_frame["Agir_intencionalmente"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bação\b|\bagente\b|\bagir\b|\batitude\b|\batividade\b|\bato\b|\bator\b|\batuar\b|\bconduz
ir\b|\bcoordenação\b|\bdesempenho\b|\bempenhar\b|\bempreender\b|\bengajar\b|\bexecução\b|\bexec
utar\b|\bfase\b|\bfazer\b|\bfeito\b|\bgesto\b|\bmedida\b|\bmissão\b|\bmovimento\b|\bobra\b|\bpasso\b|\
bperfazer\b|\bpromover\b|\brealizar\b", str(x))))
  df_frame["Agregado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacervo\b|\baglomerado\b|\bamontoado\b|\banfitrião\b|\bassembléia\b|\bbancada\b|\bb
anda\b|\bbando\b|\bbatalhão\b|\bcacho\b|\bcaravana\b|\bcardume\b|\bcasal\b|\bcírculo\ssocial\b|\bcírc
ulo\b|\bclasse\b|\bcoleção\b|\bcolônia\b|\bcombinação\b|\bcombo\b|\bcomunidade\b|\bconjunto\b|\bcor
ja\b|\bcorpo\b|\bcorporação\b|\bdupla\b|\benxame\b|\bequipe\b|\bescola\b|\besquadra\b|\besquadrão\b
|\bexército\b|\bfacção\b|\bfamília\b|\bfardo\b|\bfeixe\b|\bforça\b|\bfornada\b|\bfrota\b|\bgaláxia\b|\bgam
e\b|\bgangue\b|\bgentalha\b|\bgrupo\b|\bharém\b|\bhorda\b|\bjogo\b|\blegião\b|\blivro\b|\bmaço\b|\bmá
```

```
fia\b|\bmaioria\b|\bmanada\b|\bmassa\b|\bmatilha\b|\bmonte\b|\bmultidão\b|\bmultiplicidade\b|\bmultipli
cidade\b|\bmuvuca\b|\bninhada\b|\bpacote\b|\bpanelinha\b|\bpartido\b|\bpelotão\b|\bpenca\b|\bpilha\b|\
bplebe\b|\bpopulação\b|\bpopulacho\b|\bpunhado\b|\bquarteto\b|\bquinteto\b|\bralé\b|\brebanho\b|\bre
pertório\b|\bsafra\b|\bsexteto\b|\bsortimento\b|\btime\b|\btribo\b|\btrio\b|\btripulação\b|\btropel\b|\bturm
a\b|\buniverso\b|\bvariedade\b", str(x))))
  df_frame["Agricultura"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcultivo\b", str(x))))
  df_frame["Agrupar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bencontrar\b", str(x))))
  df_frame["Ajustar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\badaptar\b|\badequar\b",
str(x))))
  df_frame["Alcance"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdistância\b|\bvista\b",
str(x))))
  df_frame["Alimentação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baçaiteria\b|\bbar-
restaurante\b|\bbar\b|\bbarraca\b|\bbarraquinha\b|\bbirosca\b|\bbistrô\b|\bbonbonnière\b|\bboteco\b|\b
botequim\b|\bbufê\b|\bbuffet\b|\bcafé\b|\bcafeteria\b|\bcervejaria\b|\bchampanharia\b|\bchampanheria\
b|\bchocolateria\b|\bchoperia\b|\bchurrascaria\b|\bdrinkeria\b|\bfast-
food\b|\bfood\struck\b|\bhamburgueria\b|\blanchonete\b|\bloja\sde\sbebidas\salcoólicas\b|\bloja\sde\sb
ebidas\b|\bloja\sde\scervejas\b|\bmercado\b|\bmercearia\b|\bpadaria\b|\bpastelaria\b|\bpé-
sujo\b|\bpesque-
pague\b|\bpesqueiro\b|\bpizzaria\b|\bpodrão\b|\bpub\b|\brestaurante\sárabe\b|\brestaurante\sbrasileiro
\b|\brestaurante\schinês\b|\brestaurante\seuropeu\b|\brestaurante\sfrancês\b|\brestaurante\sitaliano\b|
\brestaurante\sjaponês\b|\brestaurante\smexicano\b|\brestaurante\smineiro\b|\brestaurante\sportuguê
s\b|\brestaurante\sself-service\b|\brestaurante\svegano\b|\brestaurante\b|\brodízio\b|\bself-
service\b|\bsorveteria\b|\bsupermercado\b|\btaberna\b|\btacacazeira\b|\btemakeria\b|\btrailer\b|\bveget
ariano\b", str(x))))
  df_frame["Alimentos_e_bebidas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacarajé\b|\bagnolini\b|\bágua\sde\scoco\b|\balimento\b|\bamor\sperfeito\b|\barroz-
doce\b|\bbaguete\b|\bbanana\sfrita\b|\bbarreado\b|\bbatata-
frita\b|\bbebida\salcoólica\b|\bbebida\b|\bbiscoito\b|\bbisteca\b|\bbobó\b|\bbolinho\b|\bbolo\b|\bbreja\b|\
bbrigadeiro\b|\bbruschetta\b|\bbuchada\sde\sbode\b|\bburguer\b|\bburrata\b|\bburrito\b|\bcafé\b|\bcaipi
rinha\b|\bcaipiríssima\b|\bcaipiroska\b|\bcaipisaquê\b|\bcaipivodka\b|\bcajuína\b|\bcajuzinho\b|\bcalda\
b|\bcaldeirada\b|\bcaldo\sde\scana\b|\bcaldo\b|\bcanjica\b|\bcanjiquinha\b|\bcapuccino\b|\bcarioca\b|\b
carpaccio\b|\bcaruru\b|\bcatchup\b|\bcereal\b|\bchampagne\b|\bchampanhe\b|\bcheeseburger\b|\bchi
marrão\b|\bchope\b|\bchopp\b|\bchouriço\b|\bchurrasco\b|\bchurro\b|\bcocada\b|\bcomida\scaiçara\b|\
bcomida\b|\bcompota\b|\bcoquetel\b|\bcroquete\b|\bcuca\b|\bcurau\b|\bdobradinha\b|\bdoce\b|\bdrink\
b|\bdrinque\b|\beinsbein\b|\bempada\b|\bespeciaria\b|\bespresso\b|\bexpresso\b|\bfarofa\b|\bfeijão-
tropeiro\b|\bfeijoada\b|\bfrutos\sdo\smar\b|\bgalinha\sao\smolho\spardo\b|\bgalinha\sensopada\b|\bgel
ato\b|\bgeleia\b|\bgengibre\b|\bgoiabada\b|\bgordice\b|\bguloseima\b|\bhambúrguer\b|\bhummus\b|\big
uaria\b|\bkafta\b|\bkibe\b|\bleitão\sà\spururuca\b|\blicor\b|\blimão\b|\blimonada\b|\bmaniçoba\b|\bmarg
uerita\b|\bmilkshake\b|\bmolho\b|\bmoqueca\scapixaba\b|\bmoqueca\b|\bmousse\b|\bmozzarela\b|\bn
achos\b|\bnoz-
moscada\b|\bovo\b|\bpaella\sde\smariscos\b|\bpaella\b|\bpamonha\b|\bpanqueca\b|\bpastel\b|\bpé-de-
moleque\b|\bpicadinho\b|\bpipoca\b|\bpirão\b|\bpirarucu\sde\scasaca\b|\bpizza\b|\bpodrão\b|\bpolenta\
b|\bprato\stípico\b|\bprato\b|\bpudim\b|\bquentão\b|\bquibe\b|\brabada\b|\brefeição\b|\brefrigerante\b|\
brisoto\b|\brosca\b|\bsaideira\b|\bsalada\b|\bsalgado\b|\bsalpicão\b|\bsanduíche\sde\spernil\scom\sab
acaxi\b|\bsanduíche\b|\bsarapatel\b|\bsashimi\b|\bsobrecoxa\b|\bsonho\b|\bsopa\sagnolini\b|\bsopa\b|\
bsorvete\b|\bsuco\b|\bsushi\b|\btacacá\b|\btangerina\b|\btapa\b|\btererê\b|\btorta\b|\buísque\b|\bvaca\s
atolada\b|\bvatapá\b|\bwhisky\b", str(x))))
  df_frame["Alternatividade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bem\svez\sde\b",
str(x))))
  df_frame["Alugar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\balugar\b", str(x))))
  df_frame["Alvo"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\ba\b|\bpara\b", str(x))))
  df_frame["Amalgamação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bentrelaçar\b|\bmisto\b|\bmistura\b|\bmisturar\b|\bunificado\b", str(x))))
  df_frame["Amigável_ou_hostil"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badversário\b|\binimigo\b", str(x))))
  df_frame["Andar_de_veículo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bandar\b|\bcruzeiro\b|\bfazer\smochilão\b|\bnavegação\b|\bnavegar\b|\bpegar\b|\bvelej
ar\b|\bvoar\b|\bvoo\b", str(x))))
  df_frame["Anexação_incoativa"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bencontrar\b",
str(x))))
```

```
  df_frame["Anexar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\banexar\b|\binterligar\b|\bligar\b", str(x))))
  df_frame["Animais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babelha\b|\bácaro\b|\banimal\b|\baraponga\b|\bbesouro\b|\bboi\b|\bborboleta\b|\bcachorro\b|\bcão\b|\bcarneiro\b|\bcavalo\b|\bcavaquinha\b|\bchimpanzé\b|\bcordeiro\b|\belefante\b|\bfauna\b|\bfilhote\b|\bfrango\b|\bgalinha\b|\bgalo\b|\bgato\b|\bgirafa\b|\binseto\b|\bjoaninha\b|\bleão\b|\bleitão\b|\blobo\b|\blouva-a-
deus\b|\bmacaco\b|\bmosquito\b|\bovelha\b|\bpássaro\b|\bpeixe\b|\bpeixinho\b|\bpet\b|\bpolvo\b|\bporco\b|\braia\b|\braposa\b|\bserpente\b|\bsiri\b|\btigre\b|\burubu\b|\bvaca\b|\bvaga-lume\b|\bzebra\b",
str(x))))
  df_frame["Aparato"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bandroid\b|\baplicativo\b|\bar-
condicionado\b|\bequipamento\b|\binformática\b|\bredes\ssociais\b|\bserra\b|\btecnologia\b|\busuário\b", str(x))))
  df_frame["Aparecer_em"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baparecer\b", str(x))))
  df_frame["Aplicar_calor"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdourar\b", str(x))))
  df_frame["Apoiar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bapoiar\b|\bapoio\b", str(x))))
  df_frame["Apostar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bapostar\b", str(x))))
  df_frame["Área_biológica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcerrado\b|\bdeserto\b|\bfloresta\b|\bmato\b|\boásis\b|\bpântano\b|\bpradaria\b|\bselva\b", str(x))))
  df_frame["Arma"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barma\b|\bmaça\b|\btesoura\b|\btorpedo\b", str(x))))
  df_frame["Armadilha"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\barmadilha\b", str(x))))
  df_frame["Armazenar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bguardado\b|\bmanter\b|\breservar\b", str(x))))
  df_frame["Arquitetura_de_conexão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdegrau\b|\bjanela\b|\bporta\b", str(x))))
  df_frame["Arrumar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrumado\b|\barrumar\b|\bcaprichado\b|\bequipar\b|\borganizado\b", str(x))))
  df_frame["Artefato"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\barrastão\b|\basa-
delta\b|\bbalão\b|\bbandeja\b|\bbebedouro\b|\bbico\sde\smamadeira\b|\bbolsa\b|\bbrinquedo\b|\bcadeira\b|\bcatálogo\b|\bcelular\b|\bchupeta\b|\bchuveiro\b|\bcoberta\b|\bcobertor\b|\bcomputador\b|\bconcha\b|\bcontrole\b|\bespelho\b|\bestilete\b|\bfio\b|\bfrigideira\b|\bimpressora\b|\binternet\b|\blâmina\b|\blençol\b|\blente\b|\bluva\scirúrgica\b|\bmala\b|\bmesa\b|\bmesa\b|\bmochila\b|\bmontanha\srussa\b|\borigami\b|\bpano\b|\bpapel\stoalha\b|\bpipa\b|\bplaca\b|\bpneu\b|\bprato\b|\bprisma\b|\brádio\b|\btecnologia\b|\btelefone\b|\btelevisão\b|\btesouro\b|\btoboágua\b|\btúmulo\b|\bvideo\sgame\b|\bvideo-game\b",
str(x))))
  df_frame["Artesanato"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barte\b|\bciência\b|\bcrochê\b|\bofício\b", str(x))))
  df_frame["Artes_performáticas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barte\b|\bballet\b|\bcantar\b|\bencenação\b|\bensaiar\b|\bFazer\b|\bjazz\b|\bmusical\b|\bpantomima\b|\bpeça\sde\steatro\b|\bpeça\b|\bperformance\b|\bperformar\b|\bsapateado\b|\bteatral\b|\bteatro\b|\btocar\b", str(x))))
  df_frame["Artificialidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparentemente\b|\benganosa\b", str(x))))
  df_frame["Assear"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\banti-
higiênico\b|\bbanho\b|\benxaguar\b|\blavar\b|\blavável\b|\blimpar\b", str(x))))
  df_frame["Assistência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacudir\b|\bajudar\b|\batendimento\b|\bauxílio\b|\bcuidar\b", str(x))))
  df_frame["Assistir_a_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bassistir\b|\bcomparecer\b|\bfrequentar\b|\bir\b|\bver\b", str(x))))
  df_frame["Associação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brelativo\b", str(x))))
  df_frame["Atacar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bagressor\b", str(x))))
  df_frame["Atenção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batenção\b|\batender\b|\bchamar\satenção\b|\bdar\sbola\b|\bligar\b", str(x))))
  df_frame["Atividade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batividade\b|\bbrincadeira\b|\bbrincar\b|\bdivertimento\b|\bguerrinha\b|\bjogar\b|\bpique-esconde\b|\bsensação\b", str(x))))
```

```
  df_frame["Atividades_do_turista"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bartesanato\b|\barvorismo\b|\bbanhar\b|\bbrincar\b|\bcapoeira\b|\bfrevo\b|\bpatinar\b|\b
pintura\b|\bsinuca\b|\bsurfar\b|\btirolesa\b", str(x))))
  df_frame["Atividade_em_andamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcontinuar\b|\bdecorrer\b|\bficar\b|\bpassar\b|\bprosseguir\b|\bviver\b", str(x))))
  df_frame["Atividade_iniciar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcair\b|\bcomeçar\b|\bdesencadear\b|\bentrar\b|\bestrear\b|\bgerar\b|\binauguração\b|\
binaugurar\b|\biniciante\b|\biniciar\b|\binstituir\b|\bpassar\b|\bprincipiar\b", str(x))))
  df_frame["Atividade_interromper"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdeixar\b|\bparar\b", str(x))))
  df_frame["Atividade_pausar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcortar\b|\bencerrar\b|\bimobilizar\b|\bparar\b|\breter\b", str(x))))
  df_frame["Atividade_preparada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdisponível\b|\bpreparado\b|\bpreparo\b|\bpronto\b", str(x))))
  df_frame["Atividade_preparar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bestruturar\b|\borganizar\b|\bpreparar\b|\bpreparo\b", str(x))))
  df_frame["Atividade_terminar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babdicar\b|\bacabar\b|\bconcluir\b|\bdesistir\b|\bformar\b", str(x))))
  df_frame["Atletas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badversário\b|\batleta\b|\bbateria\b|\bclube\b|\bcompetidor\b|\bdesafiante\b|\bdesportist
a\b|\bdueto\b|\bdupla\b|\bequipe\b|\besportista\b|\bjogador\b|\boponente\b|\bparaolímpico\b|\bparticipa
nte\b|\bpelotão\b|\brival\b|\bseleção\b|\btime\b|\btrio\b", str(x))))
  df_frame["Atletas_por_esporte"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamazona\b|\barqueiro\b|\barremessador\b|\batirador\b|\bboleiro\b|\bboxeador\b|\bcaia
quista\b|\bcanoísta\b|\bcarateca\b|\bcavaleiro\b|\bciclista\b|\bcorredor\b|\bdecatleta\b|\bescalador\b|\b
esgrimista\b|\bfundista\b|\bginasta\b|\bgolfista\b|\bgrequista\b|\bhalterofilista\b|\bheptatleta\b|\bjogador
\sde\sbadminton\b|\bjogador\sde\sbasquete\b|\bjogador\sde\sbeisebol\b|\bjogador\sde\sfutebol\b|\bjog
ador\sde\shandball\b|\bjogador\sde\shóquei\ssobre\sgrama\b|\bjogador\sde\spólo\b|\bjogador\sde\srú
gbi\b|\bjogador\sde\ssoftbol\b|\bjogador\sde\svôlei\b|\bjudoca\b|\blançador\b|\blevantador\b|\blutador\b
|\bmaratonista\b|\bmarchador\b|\bmeio-
fundista\b|\bmesatenista\b|\bnadador\b|\bpentatleta\b|\bpesista\b|\bpugilista\b|\bremador\b|\bsaltador\
b|\bskatista\b|\bsurfista\b|\btenista\b|\btrampoliner\b|\btrampolinista\b|\btriatleta\b|\bvelejador\b|\bveloci
sta\b", str(x))))
  df_frame["Atletas_por_posição"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babertura\b|\bala-armador\b|\bala-
pivô\b|\bala\b|\bapanhador\b|\barmador\scentral\b|\barmador\b|\barremessador\b|\bartilheiro\b|\basa\b
|\batacante\b|\bataque\b|\bavançado\b|\bbatedor\b|\bcabeça\sde\sárea\b|\bcapitão\b|\bcentral\sarmad
or\b|\bcentro\b|\bcentroavante\b|\bcontra-proa\b|\bcontra-
voga\b|\bcraque\b|\bdefensor\sexterno\b|\bdefensor\sinterno\b|\bdefensor\b|\bdefesa\scentral\b|\bdefe
sa\sdireita\b|\bdefesa\sesquerda\b|\bdefesa\b|\bentrada\sde\srede\b|\bextremo\b|\bflanqueador\b|\bfly\
shalf\b|\bfull\sback\b|\bgoleiro\b|\bhooker\b|\blançador\b|\blateral\b|\bleme\b|\blevantador\b|\blíbero\b|\
bmédio\scentral\b|\bmédio\b|\bmeia\sarmador\b|\bmeia\sdireita\b|\bmeia\sesquerda\b|\bmeia\b|\bmeio
\sde\scampo\b|\bmeio\sde\srede\b|\bmeio\sscrum\b|\bmeio-campista\b|\bmeio-
campo\b|\bmeio\b|\bnúmero\scinco\b|\bnúmero\sdois\b|\bnúmero\soito\b|\bnúmero\squatro\b|\bnúmer
o\sseis\b|\bnúmero\ssete\b|\bnúmero\strês\b|\boitavo\b|\bpassador\b|\bpilar\saberto\b|\bpilar\sfechado
\b|\bpivô\b|\bponta\sdireita\b|\bponta\sesquerda\b|\bponta\b|\bponteiro\b|\bposição\b|\bprimeira\slinha\
b|\bprimeiro\scentro\b|\bprimeiro\slateral\b|\bprimeiro\sponta\b|\bproa\b|\brebatedor\b|\brecebedor\b|\b
receptor\b|\breserva\b|\bsacador\b|\bsaída\sde\srede\b|\bsegunda\slinha\b|\bsegundo\scentro\b|\bseg
undo\slateral\b|\bsegundo\sponta\b|\bservidor\b|\bsota-proa\b|\bsota-
voga\b|\btalonador\b|\bterceira\slinha\b|\btimoneiro\b|\btitular\b|\bvoga\b|\bvolante\b|\bzaga\b|\bzaguei
ro\b", str(x))))
  df_frame["Atrair_turistas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentar\b|\batração\b|\batrair\b|\batrativo\b|\batrativo\b|\bconvidar\b|\bdestacar-
se\b|\bdestino\b|\blevar\b|\boferecer\b|\breservar\b|\bsurpreender\b", str(x))))
  df_frame["Atravessar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bascender\b|\bascensão\b|\batravessar\b|\bcircular\b|\bcruzamento\b|\bcruzar\b|\bdeci
da\b|\bdescer\b|\bmontar\b|\bpassar\b|\bpular\b|\brodear\b|\bsaltar\b", str(x))))
  df_frame["Atribuição_de_nome"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bchamar-
se\b|\bdublado\b", str(x))))
```

```
  df_frame["Atributos"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\batributo\b|\bqualidade\b",
str(x))))
  df_frame["Atributos_graduáveis"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bsuper\b",
str(x))))
  df_frame["Atributos_mensuráveis"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balto\b|\bamplo\b|\bapertado\b|\bbaixo\b|\bcaloso\b|\bcurto\b|\belevado\b|\bespesso\b|\
bestreito\b|\bfino\b|\bfundo\b|\bgrosso\b|\bleve\b|\blongo\b|\bmurcho\b|\bpesado\b|\bprofundo\b",
str(x))))
  df_frame["Auto_movimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balongamento\b|\bandar\b|\bcambalhota\b|\bcaminhada\b|\bcaminhar\b|\bcircular\b|\bc
orrer\b|\bdança\b|\bdançar\b|\bdesfilar\b|\besquentar\b|\bir\b|\bmergulhar\b|\bmovimento\b|\bnadar\b|\
bpisar\b|\bvoar\b", str(x))))
  df_frame["Avaliação_de_moralidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babsurdamente\b|\babsurdo\b|\bantiético\b|\bbaixo\b|\bbom\b|\bcanalha\b|\bcandongu
eiro\b|\bcerto\b|\bdegenerado\b|\bdepravação\b|\bdepravado\b|\bdescente\b|\bdesonroso\b|\bdoloso\b
|\berrado\b|\berrar\b|\berro\b|\bescuro\b|\bético\b|\bgeneroso\b|\bhorroroso\b|\bimoral\b|\bimpróprio\b|\
binescrupuloso\b|\biníquo\b|\binsidioso\b|\bíntegro\b|\bjusto\b|\bmaldoso\b|\bmau\b|\bmelhor\b|\bmeno
s\b|\bmoral\b|\bnefasto\b|\bobsceno\b|\bpecaminoso\b|\bpecar\b|\bperverso\b|\bpior\b|\bréprobo\b|\bre
pulsivo\b|\bvil\b|\bvirtuoso\b", str(x))))
  df_frame["Avaliar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bachar\b|\bavaliação\b|\bavaliar\b|\bbom\b|\bbom\b|\bimportar\b|\bjulgamento\b|\bjulga
r\b|\blamentável\b|\bmaravilhoso\b|\bmelhor\b", str(x))))
  df_frame["Boa_vontade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bboa-
vontade\b|\bdispor\b", str(x))))
  df_frame["Caçar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcaça\b|\bcaçada\b|\bcaçador\b|\bcaçar\b|\bpescar\b", str(x))))
  df_frame["Cair_no_sono"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badormecer\b|\bdesmaiar\b", str(x))))
  df_frame["Campos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagropecuária\b|\bâmbito\b|\barquitetura\b|\barte\b|\bartes\svisuais\b|\bartístico\b|\bast
rofísica\b|\bastrofísico\b|\bastrologia\b|\bastronomia\b|\baviação\b|\bcampo\b|\bciência\b|\bcientífico\b|
\bcosmológico\b|\bcrítica\b|\bculinária\b|\bcultura\b|\bdança\b|\bdemografia\b|\bdesenho\b|\bdisciplina\
b|\bdomínio\b|\bdrama\b|\becologia\b|\beconomia\b|\bfilosofia\b|\bfinança\b|\bfísica\b|\bgastronomia\b|
\bgeografia\b|\bhistória\b|\bhumanas\b|\bindustrialização\b|\binglês\b|\blazer\b|\blíngua\b|\bmatemátic
a\b|\bmorfologia\b|\bmúsica\b|\bpoesia\b|\bquântico\b|\brubrica\b|\bsemântica\b|\btelecomunicação\b",
str(x))))
  df_frame["Caos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbagunça\b|\bbagunçado\b|\bturbulento\b", str(x))))
  df_frame["Capacidade_ação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baptidão\b|\bapto\b|\bcapacidade\b|\bcapacitar\b|\bcapaz\b|\bcompetência\b|\bconseg
uir\b|\bdar\b|\bdom\b|\bhabilidade\b|\bimpotente\b|\bincapaz\b|\bpoder\b|\btalento\b", str(x))))
  df_frame["Careza"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacessibilidade\b|\bacessível\b|\bbaixo\scusto\b|\bbarato\b|\bcaro\b|\bcustar\b|\bcusto\
b|\bdespesa\b|\bexorbitante\b|\bgratuito\b|\boneroso\b|\bsuperfaturado\b|\bvaler\b", str(x))))
  df_frame["Catástrofe"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcrise\b|\bfatalidade\b|\bincidente\b", str(x))))
  df_frame["Categorização"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bclassificação\b|\bclassificado\b|\bclassificar\b|\bconsiderado\b|\bconsiderar\b|\bdeclar
ar\b|\binterpretar\b|\breconhecer\b", str(x))))
  df_frame["Causalidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bassim\b|\bcausa\b|\bcausar\b|\bconsequência\b|\bconsequentemente\b|\bculminar\b|\
bdar\b|\bde\smodo\sque\b|\bdeixar\b|\bdesencadear\b|\bdespertar\b|\bdever\b|\befeito\b|\bentão\b|\bfa
zer\scom\sque\b|\bfazer\b|\bmedida\b|\bpor\b|\bporque\b|\bportanto\b|\bprovocar\b|\brender\b|\brespo
nsável\b|\bresultado\b|\bresultado\b|\bresultar\b|\btornar\b", str(x))))
  df_frame["Causar_acordar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bacordar\b", str(x))))
  df_frame["Causar_continuar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacalentar\b|\bpreservar\b", str(x))))
  df_frame["Causar_dano"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacertar\b|\bapedrejar\b|\barranhar\b|\bbater\b|\bferir\b|\bmachucar\b|\btorcer\b",
str(x))))
```

```
  df_frame["Causar_emoção"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdeixar\b", str(x))))
  df_frame["Causar_estar_incluído"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bincluir\b",
str(x))))
  df_frame["Causar_expansão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bampliar\b|\baumentar\b|\bcaprichar\b|\bcrescimento\b|\bdiminuir\b|\besticar\b|\bexpan
dir\b|\bminimizar\b", str(x))))
  df_frame["Causar_fazer_progresso"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdedicar\b|\besmerar\b|\binvestimento\b|\binvestir\b|\bsofisticar\b", str(x))))
  df_frame["Causar_ficar_afiado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bafiar\b",
str(x))))
  df_frame["Causar_ficar_molhado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmolho\b",
str(x))))
  df_frame["Causar_ficar_seco"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\benxugar\b|\bsecar\b", str(x))))
  df_frame["Causar_fragmentar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrebentar\b|\bquebrar\b|\bromper\b", str(x))))
  df_frame["Causar_fundir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcombinar\b|\bgrupo\b|\bjuntar\b|\breunir\b", str(x))))
  df_frame["Causar_movimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagitar\b|\barejar\b|\batrair\b|\bempurrar\b|\bjogar\b|\blançar\b|\blargar\b|\blevantar\b|\b
movimentar\b|\bsacar\b|\bsubir\b|\btampar\b", str(x))))
  df_frame["Causar_movimento_fluídico"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bentornar\b", str(x))))
  df_frame["Causar_mudança"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balterar\b|\bcustomizado\b|\bmodificador\b|\bmudar\b|\btransformar\b|\btrocar\b",
str(x))))
  df_frame["Causar_mudança_de_força"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\breforçar\b", str(x))))
  df_frame["Causar_mudança_de_posição_em_uma_escala"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\breduzir\b|\bvalorizar\b", str(x))))
  df_frame["Causar_mudança_de_temperatura"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\brefrigerar\b", str(x))))
  df_frame["Causar_mudar_de_lugar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchacoalhar\b", str(x))))
  df_frame["Causar_perceber"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapontar\b|\bapresentação\b|\bapresentar\b|\bassinalar\b|\bdemonstrar\b|\besbanjar\b|
\bexpor\b|\bexposição\b|\biluminar\b|\blançar\b|\bmostrar\b|\bpublicar\b|\brepresentar\b|\brevelar\b",
str(x))))
  df_frame["Causar_retomar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\breviver\b", str(x))))
  df_frame["Causar_terminar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdissipar\b|\bterminar\b", str(x))))
  df_frame["Ceder"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bentregar\b|\bimplacavelmente\b", str(x))))
  df_frame["Cenário_da_história"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brelembrar\b",
str(x))))
  df_frame["Cenário_de_aquisição"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baquisição\b",
str(x))))
  df_frame["Cenário_de_doação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcontribuir\b|\bcortesia\b", str(x))))
  df_frame["Cenário_de_importação_e_exportação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexportador\b", str(x))))
  df_frame["Cenário_de_interação_médica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacidentado\b|\bcirurgia\b|\bcirúrgico\b|\bdegenerativo\b|\bdificuldade\b|\bponto\b|\bvíti
ma\b", str(x))))
  df_frame["Cenário_de_obrigação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdever\b",
str(x))))
  df_frame["Cenário_do_comércio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcobrar\b|\bcomércio\b|\bdesconto\b|\bpreço\b|\bserviço\b|\btarifa\b", str(x))))
  df_frame["Cenário_do_turismo"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bturismo\b",
str(x))))
```

```
  df_frame["Cenário_do_turismo_estada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bestada\b|\bestadia\b|\bestar\b", str(x))))
  df_frame["Cenário_do_turismo_partida"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bida\b|\bir\sembora\b|\bpartida\b|\bpartir\b|\bsair\b", str(x))))
  df_frame["Cenário_visita_chegada"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bChega\b",
str(x))))
  df_frame["Cercanias"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bao\sredor\sde\b|\barredor\b|\bcercar\b|\bcircundar\b|\benvolto\b|\bpor\b|\bredondeza\b
|\bredor\b|\brodeado\b", str(x))))
  df_frame["Cerimônias"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babertura\b|\bcerimônia\b|\bencerramento\b|\bmedalha\b", str(x))))
  df_frame["Certeza"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bassegurar\b|\bcertamente\b|\bdecerto\b|\bdúvida\b|\benigmático\b|\bexatamente\b|\bi
ncerteza\b|\bmistério\b|\bmisterioso\b", str(x))))
  df_frame["Chance"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bimpossível\b|\bpossível\b|\btalvez\b|\btender\b", str(x))))
  df_frame["Chegada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baportar\b|\bchegada\b|\bchegar\b", str(x))))
  df_frame["Chegada_ao_alojamento"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcheck-
in\b", str(x))))
  df_frame["Chegada_ao_destino"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdesembarcar\b|\bdesembarque\b", str(x))))
  df_frame["Chegar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparecer\b|\baportar\b|\baproximar\b|\bchegar\b|\bentrar\b|\bregressar\b|\bretornar\b|\
bvir\b|\bvoltar\b", str(x))))
  df_frame["Chegar_a_acreditar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchutar\b|\bconclusão\b", str(x))))
  df_frame["Circunstâncias_contrárias"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapesar\sde\b|\bmesmo\sque\b", str(x))))
  df_frame["Classificação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bgraduação\b", str(x))))
  df_frame["Classificação_biológica"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bespécie\b",
str(x))))
  df_frame["Clima"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bártico\b|\bavalanche\b|\bcerração\b|\bclima\b|\bdilúvio\b|\benchente\b|\benxurrada\b|\
bgeada\b|\binundação\b|\bnévoa\b|\bonda\b|\bressaca\b|\bseco\b|\bsol\b|\btempestade\b|\btropical\b|\
búmido\b|\bvendaval\b", str(x))))
  df_frame["Codificar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexpressão\b|\bfrase\b|\bpalavra\b", str(x))))
  df_frame["Cogitação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcismar\b|\bconcentrar\b|\bcontemplação\b|\bcontemplativo\b|\blevar\sem\sconta\b|\bp
ensamento\b|\bpensar\b|\bponderar\b|\brepensar\b|\bvir\sà\smente\b", str(x))))
  df_frame["Coincidência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcasualidade\b|\bcoincidentemente\b", str(x))))
  df_frame["Colaboração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcolaborar\b|\binteração\b", str(x))))
  df_frame["Colocação_espacial"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\blá\b", str(x))))
  df_frame["Colocação_temporal"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bà\smedida\sque\b|\ba\b|\bagora\b|\bantigamente\b|\bantigo\b|\bao\slongo\sde\b|\batu
al\b|\batualmente\b|\bdentro\sde\b|\bdurante\b|\bem\b|\benquanto\b|\bentão\b|\bfuturo\b|\bfuturo\b|\bh
oje\sem\sdia\b|\bhoje\b|\bimediatamente\b|\bmais\b|\bmoderno\b|\bpor\svolta\sde\b|\bpor\b|\bpré-
histórico\b|\bquando\b|\bquando\b|\brecentemente\b|\btão\slogo\b|\búltimamente\b", str(x))))
  df_frame["Colocar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balinhamento\b|\baplicar\b|\bcolocar\b|\bestacionar\b|\blevar\b|\bmergulhar\b|\bparar\b|
\bpendurar\b|\bpõem\b|\bpôr\b", str(x))))
  df_frame["Colonização"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcolonizar\b|\binstalar\b", str(x))))
  df_frame["Comércio_comprar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badquirir\b|\bcliente\b|\bcompra\b|\bcomprar\b|\bconsumidor\b", str(x))))
  df_frame["Comércio_pagar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcouvert\b|\bimposto\b|\bpagamento\b", str(x))))
```

```
  df_frame["Comércio_receber"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcobrar\b",
str(x))))
  df_frame["Comércio_vender"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomercializar\b|\bleilão\b|\bpromoção\b|\bvenda\b|\bvender\b", str(x))))
  df_frame["Comissão_técnica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\banalista\sde\sdesempenho\b|\bauxiliar\stécnico\b|\bauxiliar\b|\bchef\b|\bcomissão\sté
cnica\b|\bcoordenador\b|\bcozinheiro\b|\bdiretor\b|\bfisiologista\b|\bfisioterapeuta\b|\bfotógrafo\b|\bger
ente\b|\binstrutor\b|\bmassagista\b|\bmédico\b|\bnutricionista\b|\bobservador\stécnico\b|\bolheiro\b|\bp
reparador\sde\sgoleiro\b|\bpreparador\sfísico\b|\bpsicólogo\b|\broupeiro\b|\bsegurança\b|\bsupervisor\
b|\btécnico\b|\btreinador\sassistente\b|\btreinador\b|\bveterinário\b", str(x))))
  df_frame["Comparação_avaliativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomparar\b|\bdo\sque\b|\bequivaler\b|\bigualmente\b|\bincomparável\b|\blonge\b|\bm
ais\b|\bmelhor\b|\bmelhorar\b|\bmenor\b|\bpiorar\b", str(x))))
  df_frame["Comparecer"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bauto-
atendimento\b|\bir\b", str(x))))
  df_frame["Compatibilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcondizente\b|\bcondizer\b|\bconsistência\b|\bharmonia\b", str(x))))
  df_frame["Competição"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbrigar\b|\bcampeonato\b|\bcombate\b|\bcompetição\b|\bcompetidor\b|\bcompetir\b|\bc
ompetitivo\b|\bconcorrência\b|\bdesafio\b|\bdisputa\b|\bdisputar\b|\bencarar\b|\bgame\b|\bjogar\b|\bjog
o\b|\bliga\b|\brivalidade\b|\btorneio\b", str(x))))
  df_frame["Completude"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomplementar\b|\bcompletar\b|\bcompleto\b|\btotal\b|\btotalidade\b", str(x))))
  df_frame["Complexidade_sistêmica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomplexidade\b|\bsimples\b", str(x))))
  df_frame["Comprar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomprar\b|\bcompras\b|\bcusto\sbenefício\b", str(x))))
  df_frame["Comprometimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bameaçar\b|\bjuramento\b|\bjurar\b|\bprometer\b", str(x))))
  df_frame["Comunicação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomunicar\b|\btransmitir\b", str(x))))
  df_frame["Comunicação_de_julgamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcelebrar\b|\bcrítica\b|\bcriticar\b|\bcrítico\b", str(x))))
  df_frame["Comunicação_direta_de_julgamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bobrigado\b|\bobrigado\b|\bparabéns\b", str(x))))
  df_frame["Comunicação_resposta"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\breplicar\b|\bresponder\b|\btornar\b", str(x))))
  df_frame["Comunicar_categorização"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdefinição\b|\bdefinir\b|\bdeterminação\b|\bdeterminado\b|\bretratar\b|\bsimbolizar\b",
str(x))))
  df_frame["Concessão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bainda\sassim\b|\bainda\sque\b|\bapesar\sde\b|\bexceção\b|\bmas\b|\bna\srealidade\b|
\bna\sverdade\b|\bno\sentanto\b", str(x))))
  df_frame["Condições_médicas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badoecer\b|\balérgico\b|\bárea\b|\bcadeirante\b|\bcadeirante\b|\bcâncer\b|\bcardíaco\b|
\bcirúrgico\b|\bdeficiência\b|\bderrame\b|\bdiagnosticado\b|\bdistrofia\b|\bdoença\b|\bdoente\b|\bdoer\
b|\bdor\b|\bepidemia\b|\besclerose\slateral\samiotrófica\b|\bfratura\b|\bgrávida\b|\bhemorragia\b|\binter
nado\b|\blesão\b|\bnervoso\b|\bpaciente\b|\bparada\srespiratória\b|\bpassar\smal\b|\bpatogénico\b|\bp
ortador\b|\bproblema\srespiratório\b|\bproblema\b|\breceber\salta\b|\bsaúde\b|\bvítima\b|\bvômito\b",
str(x))))
  df_frame["Conduta"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcomportamento\b", str(x))))
  df_frame["Conectores"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcabo\b|\bcorda\b|\bfilamento\b|\bfita\sadesiva\b|\bgancho\b|\bluva\b", str(x))))
  df_frame["Conexão_cognitiva"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bassociação\b|\bassociado\b|\bconectar\b|\benvolver\b|\bligar\b|\bremontar\b|\bter\sa\s
ver\b", str(x))))
  df_frame["Confiar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bconfiar\b", str(x))))
  df_frame["Confrontar_problema"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bencarar\b|\benfrentar\b|\bpassar\b", str(x))))
```

```
  df_frame["Conhecimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacessível\b|\bachar\b|\bcompreender\b|\bconcepção\b|\bconhecer\b|\bconhecimento\
b|\bconsiderar\b|\bcrer\b|\bdesavisado\b|\bdiscernimento\b|\bentender\b|\bfazer\sideia\b|\bideia\b|\bim
aginação\b|\bimaginar\b|\binacessível\b|\binalcançável\b|\bnoção\b|\bpensamento\b|\bpensar\b|\brepe
nsar\b|\bsabedoria\b|\bsaber\b|\bsuspeitar\b|\bter\b", str(x)))
  df_frame["Conquistar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconquistar\b|\btomar\sconta\b|\btomar\b", str(x))))
  df_frame["Construir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconceber\b|\bconstruir\b|\berguer\b|\binaugurar\b|\breforma\b|\breformado\b|\brefor
mar\b|\bresidência\b", str(x))))
  df_frame["Contatar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchamar\b|\bcontactar\b|\bcontato\b|\bcorresponder\b|\bligar\b|\btelefonar\b", str(x))))
  df_frame["Conter"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\balojar\b|\bter\b", str(x))))
  df_frame["Contingência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdependência\b|\bdepender\b|\bindependente\b", str(x))))
  df_frame["Contra-atacar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcontra-
atacar\b|\bcontra-ataque\b", str(x))))
  df_frame["Contratar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bterceirizar\b", str(x))))
  df_frame["Contrição"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\barrepender-
se\b|\barrepender\b|\barrependido\b|\barrependimento\b|\bcontrição\b|\bcontrito\b|\bculpa\b|\bculpado\
b|\bdesculpa\b|\bdesculpar\b|\bimpenitente\b|\bpenalizado\b|\bpenitência\b|\bpenitente\b|\bremorso\b|\
bremorso\b", str(x))))
  df_frame["Controlar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcondicionar\b|\bdeterminar\b|\bregulamentação\b|\bregulamentar\b|\bregulamento\b",
str(x))))
  df_frame["Conversar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bbate-
papo\b|\bcontar\b|\bconversar\b|\bpiada\b|\bzoar\b", str(x))))
  df_frame["Convidado_e_anfitrião"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconvida\b|\bconvidado\b", str(x))))
  df_frame["Cor"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balaranjado\b|\bamarelado\b|\bamarelo\b|\bazul\b|\bbranco\b|\bcolorido\b|\bcor\b|\bpret
o\b|\bverde-clara\b|\bverde\b|\bvermelho\b|\bvioleta\b", str(x))))
  df_frame["Cortar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcortar\b|\bcorte\b|\bpicadinha\b|\bpicado\b|\btosa\b", str(x))))
  df_frame["Costume"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacostumar\b|\bclássico\b|\bcostumar\b|\bcostume\b|\bparadigma\b|\btradição\b|\btradi
cional\b", str(x))))
  df_frame["Cotema"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconduzir\b|\bguiar\b|\bseguir\b", str(x))))
  df_frame["Crença_religiosa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcredo\b|\bcrença\b|\bcrer\b|\bdevoto\b|\bfé\b|\bfiel\b|\breligião\b", str(x))))
  df_frame["Criação_culinária"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacrescentar\b|\badicionar\b|\bassado\b|\bassar\b|\bbater\b|\bcolocar\b|\bconsertar\b|\
bcozinhar\b|\bcozinheiro\b|\bculinária\b|\bculinário\b|\bdecorar\b|\bdegustação\b|\bdeixar\b|\bdespejar\
b|\bdourar\b|\bfazer\b|\bfeito\b|\bfritar\b|\bfrito\b|\bfritura\b|\bgratinar\b|\bgrelhar\b|\binventar\b|\bmexer\
b|\bmilanesa\b|\bparmegiana\b|\bpiamontese\b|\bpicado\b|\bpolvilhar\b|\bpreparação\b|\bpreparar\b|\b
salgar\b|\btemperar\b", str(x))))
  df_frame["Criar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconceber\b|\bconsistir\b|\bcriar\b|\bformação\b|\bformar\b|\binovação\b|\binovar\b|\bin
stituir\b|\bproduzir\b", str(x))))
  df_frame["Criar_arte_física"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bartista\b|\bdesenhar\b|\bescalar\b|\besculpir\b|\bpintado\b|\bpintar\b|\btirar\sfoto\b",
str(x))))
  df_frame["Criar_intencionalmente"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barmação\b|\barmar\b|\bconfigurar\b|\bcriar\b|\bdar\sorigem\b|\belaborar\b|\bestabelec
er\b|\bfazer\b|\bfundado\b|\bfundar\b|\bideia\b|\bInventa\b|\bpreparar\b|\bprodutor\b|\bproduzir\b|\breali
zar\b|\bter\b", str(x))))
  df_frame["Criar_representação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdesenhar\b|\besboçar\b|\besculpir\b|\bfoto\b|\bfotografar\b|\bfotografia\b|\bilustrado\b|\
bpintar\b", str(x))))
```

```
  df_frame["Criminalidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcrime\b", str(x))))
  df_frame["Cultivar_alimentos"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcultivar\b",
str(x))))
  df_frame["Cumprimento_de_normas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcontrariar\b|\bfiel\b|\bmandar\b|\bobedecer\b|\bseguir\b", str(x))))
  df_frame["Cura"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btratamento\b", str(x))))
  df_frame["Dançar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bsambar\b", str(x))))
  df_frame["Danificar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrebentar\b|\bfurar\b|\brasgar\b|\btrincar\b", str(x))))
  df_frame["Dar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbrinde\b|\bceder\b|\bdádiva\b|\bdar\b|\bdoação\b|\bprenda\b|\bpresente\b|\bsouvenir\
b", str(x))))
  df_frame["Dar_à_luz"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdar\sorigem\b", str(x))))
  df_frame["Dar_forma"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btorcer\b", str(x))))
  df_frame["Dar_impressão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparentar\b|\baparente\b|\bcheirar\b|\bfeder\b|\bimpressão\b|\blembrar\b|\bparecer\b|\
bprovar\b|\bsoar\b", str(x))))
  df_frame["Data_comemorativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baniversário\b|\bcarnaval\b|\bNatal\b", str(x))))
  df_frame["Decidir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdecidir\b|\bdecisão\b|\bdecisiva\b|\bestabelecer\b|\bresolver\b", str(x))))
  df_frame["Declaração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badmitir\b|\bafirmação\b|\bafirmar\b|\balegação\b|\balegar\b|\bamuar\b|\banunciar\b|\b
anúncio\b|\barriscar\b|\batestar\b|\bcitar\b|\bcomentar\b|\bcomentário\b|\bcompletar\b|\bcomprovar\b|\
bconcessão\b|\bconfessar\b|\bconfirmar\b|\bconfissão\b|\bconjetura\b|\bconjeturar\b|\bcontar\b|\bconta
r\b|\bconversa\b|\bconversar\b|\bdeclaração\b|\bdeclarar\b|\bdescrever\b|\bdetalhar\b|\bdizer\b|\bescla
recimento\b|\bescrever\b|\bexclamação\b|\bexclamar\b|\bexplicação\b|\bexplicar\b|\bexplicar\b|\bexpre
ssar\b|\bexultar\b|\bfala\b|\bfalar\b|\binsistência\b|\binsistir\b|\bmanter\b|\bmenção\b|\bmencionar\b|\b
mensagem\b|\bnegação\b|\bnotar\b|\bobservar\b|\borar\b|\bousar\b|\bpremissa\b|\bprestar\sconta\b|\b
proclamação\b|\bproclamar\b|\bprofessar\b|\bpromulgação\b|\bpronunciamento\b|\bpronunciar\b|\bpro
por\b|\bproposição\b|\bproposta\b|\breafirmar\b|\breclamar\b|\brefutar\b|\breiterar\b|\brelacionar\b|\brel
atar\b|\brelato\b|\brelatório\b|\brepetir\b|\breproduzir\b|\bser\scomo\b|\bsermão\b|\bsorrir\b|\bsugerir\b",
str(x))))
  df_frame["Degustar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdegustar\b|\bdeliciar-
se\b|\bexperimentar\b|\bprovar\b", str(x))))
  df_frame["Deixado_por_fazer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdeixado\b|\bdeixar\b|\brestante\b|\brestar\b|\bsobrar\b", str(x))))
  df_frame["Deixar_de_ser"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdesaparecer\b",
str(x))))
  df_frame["Delegação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconfederação\b|\bdelegação\b", str(x))))
  df_frame["Delitos"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bfraude\b", str(x))))
  df_frame["Desastre_natural"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bavalanche\b|\bciclone\b|\bdesastre\b|\bdesertificação\b|\bmaremoto\b|\bseca\b|\bterre
moto\b", str(x))))
  df_frame["Descrição_corporal_holística"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfeminino\b", str(x))))
  df_frame["Descrição_de_duração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbreve\b|\bcontínuo\b|\bcrônico\b|\bcurto\b|\bduradouro\b|\bdurável\b|\befêmero\b|\bes
tendido\b|\beternamente\b|\beterno\b|\bfase\b|\binterino\b|\blongo\b|\bmomentâneo\b|\bperpétuo\b|\bp
ersistente\b|\brápido\b|\bsustentável\b", str(x))))
  df_frame["Descrição_parte_do_corpo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bemagrecer\b|\bgorda\b|\bliso\b", str(x))))
  df_frame["Descrição_químico-sensorial"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcheiro\b|\bcheiroso\b|\bcrocante\b|\bdelicioso\b|\bdoce\b|\bsalgado\b|\btorrada\b",
str(x))))
  df_frame["Desejabilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badmirável\b|\baprovado\b|\barrasar\b|\bbabaca\b|\bbem-
cuidado\b|\bbem\b|\bbenigno\b|\bbobo\b|\bbom\b|\bchato\b|\bchato\b|\bcorrupto\b|\bdaora\b|\bdecente
\b|\bdeprimente\b|\bdesagradável\b|\bdesejável\b|\bdeslumbrante\b|\bdespojado\b|\bdespreparado\b|\
```

```
bdigno\b|\bdisponível\b|\bdoce\b|\beclético\b|\beficiente\b|\belitizado\b|\bespetacular\b|\besplêndido\b|\
bestupendo\b|\bexcelência\b|\bexcelente\b|\bexcepcional\b|\bexecrável\b|\bextraordinário\b|\bextremo
\b|\bexuberante\b|\bfabuloso\b|\bfantástico\b|\bfavorável\b|\bfenomenal\b|\bferrado\b|\bformidável\b|\b
horrível\b|\bidílico\b|\bimundo\b|\bincrível\b|\bindescritível\b|\bIndistinguível\b|\binfeliz\b|\binferior\b|\bin
útil\b|\binvasivo\b|\birresistível\b|\bjoia\b|\bjulgar\b|\bjusto\b|\blamentável\b|\bleve\b|\blimpo\b|\blixo\b|\b
magnífico\b|\bmaravilha\b|\bmaravilhoso\b|\bmedíocre\b|\bmeia-
boca\b|\bmelhor\b|\bmerda\b|\bmetido\b|\bmiserável\b|\bnormal\b|\bnovo\b|\bótimo\b|\bouro\b|\bpatétic
o\b|\bperdido\b|\bperito\b|\bpéssimo\b|\bpior\b|\bpobre\b|\bpodre\b|\bpopular\b|\bporcaria\b|\bprimoros
o\b|\brazoável\b|\bruim\b|\bsaudável\b|\bsensacional\b|\bsimples\b|\bsofisticado\b|\bsofrível\b|\bsujo\b|
\bsuper\b|\bsupremo\b|\bsurreal\b|\bterrível\b|\btolerável\b|\btremendo\b|\bvelho\b|\bverdadeiramente\
b|\bverdadeiro\b|\bviolento\b", str(x))))
  df_frame["Desejar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balmejar\b|\bambição\b|\bambicionar\b|\bambicioso\b|\banseio\b|\bânsia\b|\bansiar\b|\b
ansioso\b|\baspiração\b|\baspirar\b|\bcobiça\b|\bcobiçar\b|\bdefinhar\b|\bdesejado\b|\bdesejar\b|\bdes
ejo\b|\bdesejoso\b|\bdeterminação\b|\besperança\b|\besperar\b|\bfenômeno\b|\bimpaciente\b|\bimpera
tivo\b|\bimpulso\b|\binteressado\b|\bluxúria\b|\bprocurar\b|\bquerer\b|\bquerer\b|\brelutante\b|\bsaudad
e\b|\bsede\b|\bsedento\b|\bvontade\b", str(x))))
  df_frame["Desembarcar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdesembarcação\b|\bdesembarcar\b|\bdesmontar\b|\bpousar\b", str(x))))
  df_frame["Deslocamento_intencional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baceder\b|\bescalar\b|\bsubir\b", str(x))))
  df_frame["Deslocar-se"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbotecar\b|\bpassar\b|\bpassear\b|\bpasseio\b", str(x))))
  df_frame["Despedaçar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bquebrar\b", str(x))))
  df_frame["Destacar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdescolar\b", str(x))))
  df_frame["Destruir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdestruição\b", str(x))))
  df_frame["Diferenciação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdiferente\b|\bdistinção\b|\bdistinguir\b", str(x))))
  df_frame["Dificuldade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomplexo\b|\bcrítico\b|\bdifícil\b|\bdificuldade\b|\bfácil\b|\bfacilidade\b|\bfacilmente\b|\
bimpenetrável\b|\bimpossível\b|\bproblema\b", str(x))))
  df_frame["Dificultar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batraso\b|\bdemorar\b|\bdificultar\b", str(x))))
  df_frame["Dimensão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baltura\b|\bárea\b|\bcomprimento\b|\bnível\b", str(x))))
  df_frame["Dinamismo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdinâmico\b|\bintensidade\b|\bintenso\b|\bpreguiçoso\b|\bvibrante\b", str(x))))
  df_frame["Dinheiro"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcartão\sde\scrédito\b|\bcartão\b|\bdinheiro\b|\bnota\b", str(x))))
  df_frame["Direção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badiante\b|\balto\b|\balto\b|\bbaixo\b|\bcaminho\b|\bcima\b|\bdireção\b|\bdireita\b|\besq
uerda\b|\bfora\b|\bleste\b|\bleste\b|\bnorte\b|\bnorte\b|\boeste\b|\boeste\b|\bpara\scima\b|\bpara\scima
\b|\bpara\sfrente\b|\brumo\b|\bsul\b|\bsul\b", str(x))))
  df_frame["Discussão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconferência\b|\bconvenção\b|\bconversa\b|\bdebate\b|\bdiscurso\b|\bpainel\b|\bpalest
ra\b|\bplenária\b|\breunião\b|\bseminário\b", str(x))))
  df_frame["Discutir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bargumento\b|\blutar\b|\bprotesto\b", str(x))))
  df_frame["Dispersão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdifundido\b|\bdifundir\b|\bdifuso\b|\bdispersão\b|\bdispersar\b|\bdissolver\b|\bdistribuiç
ão\b|\bdistribuir\b|\bespalhar\b", str(x))))
  df_frame["Distinção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparência\b|\baspecto\b|\bcaracterístico\b|\bdiferenciar\b|\bdistinção\b|\bgarantir\b|\b
marcado\b|\bmarcar\b|\bter\b", str(x))))
  df_frame["Diversidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamplo\b|\bdiversidade\b|\bdiversificado\b|\bdiverso\b|\bextensão\b|\bheterogeneidade
\b|\bheterogêneo\b|\bhomogeneidade\b|\bhomogêneo\b|\blargura\b|\bmistura\b|\bmultifacetada\b|\bmu
ltifacetado\b|\bmultiplicidade\b|\bmúltiplo\b|\bpuro\b|\bsortido\b|\bsortimento\b|\buniforme\b|\buniformid
ade\b|\bvariabilidade\b|\bvariação\b|\bvariado\b|\bvariedade\b|\bvário\b", str(x))))
```

```
  df_frame["Divisão_temporal_do_esporte"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacréscimo\b|\bassalto\b|\bdisputa\sde\spênaltis\b|\bfinal\b|\bgolden\sscore\b|\binício\b
|\bintervalo\b|\bprorrogação\b|\bquarto\b|\brodada\b|\brotina\b|\bround\b|\bsérie\b|\bset\b|\btempo\sreg
ulamentar\b|\btempo\b|\btentativa\b|\bvolta\b", str(x))))
  df_frame["Dizer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bavisar\b|\bcontar\b|\bdesabafar\b|\bdizer\b|\bfalar\b|\bnarrar\b", str(x))))
  df_frame["Documentos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacordo\b|\bautorização\b|\bcarta\b|\bcomprovante\sde\svacinação\b|\bconcessão\b|\b
confirmação\b|\bcontrato\b|\bcontratual\b|\bconvocação\b|\bcupom\sfiscal\b|\bdecisão\b|\bdeclaração\
b|\bdepoimento\b|\bdescoberta\b|\bdiploma\b|\bdireito\b|\bdocumentação\b|\bdocumento\b|\bescritura\
b|\bgarantia\b|\bidentificação\b|\bintimação\b|\blei\b|\blicença\b|\bnota\b|\bopinião\b|\bordem\b|\bpapéi
s\b|\bpassaporte\b|\bpermissão\b|\bsumário\b|\btestamento\b|\btestemunho\b|\btítulo\b|\btratado\b|\bvi
sto\b", str(x))))
  df_frame["Doença"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcâncer\b|\bdoença\b|\bhérnia\b|\bzika\b", str(x))))
  df_frame["Dominar_situação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdominar\b|\bpredominar\b", str(x))))
  df_frame["Domínio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barquitetônico\b|\bcientificamente\b|\bcultural\b|\bem\stermos\b|\bhistoricamente\b|\bhi
stórico\b|\bmusical\b|\bpsicológico\b|\bsocial\b", str(x))))
  df_frame["Dormir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdormir\b|\binconsciente\b|\bsono\b", str(x))))
  df_frame["Duplicação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bclonado\b", str(x))))
  df_frame["Eclipse"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamortalhado\b|\bamortalhar\b|\bapagar\b|\bblindado\b|\bblindar\b|\bbloquear\b|\bcobe
rto\b|\bcobrir\b|\beclipse\b|\beclipse\b|\bencoberto\b|\bencobrir\b|\besconder\b|\bescondido\b|\bmascar
ado\b|\bmascarar\b|\bobscurecer\b|\bobscurecido\b|\bobstruir\b|\boclusão\b|\bocultação\b|\boculta r\b|\
bproteger\b|\bprotegido\b|\bvelado\b|\bvelar\b", str(x))))
  df_frame["Economia"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\beconomia\b|\beconômico\b", str(x))))
  df_frame["Educação_ensino"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacadêmico\b|\balfabetização\b|\baluno\b|\baprender\b|\baprendizado\b|\baula\b|\bbac
harelado\b|\bcursar\b|\bcurso\b|\bdiplomar\b|\bdisciplina\b|\bdoutorado\b|\beducação\b|\beducacional\
b|\beducado\b|\beducar\b|\bensinamento\b|\bensinar\b|\bentender\b|\blecionar\b|\bmagistério\b|\bmate
mática\b|\bmestrado\b|\bnormalista\b|\bprofessor\b|\bregistro\b|\buniversitário\b", str(x))))
  df_frame["Eletroeletrônicos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bar\scondicionado\b|\bfotocopiadora\b|\bmáquina\sde\slavar\b|\bmáquina\b|\bprancha\
b", str(x))))
  df_frame["Emergência"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bemergência\b", str(x))))
  df_frame["Emitir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bre-emitir\b", str(x))))
  df_frame["Emoção_com_foco_no_experienciador"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babominar\b|\badoração\b|\badorar\b|\badorável\b|\bagradecer\b|\balegre\b|\balegrem
ente\b|\bamar\b|\bamor\b|\bantipatia\b|\bapaixonar\b|\bapiedar\b|\bapreensivo\b|\barrepender\b|\barre
pendimento\b|\baversão\b|\bboquiaberto\b|\bcalmo\b|\bcarinho\b|\bcarinhosamente\b|\bchateado\b|\bc
heio\b|\bcompaixão\b|\bconforto\b|\bconsolação\b|\bdeliciar\b|\bdesconforto\b|\bdesesperado\b|\bdese
sperar\b|\bdesespero\b|\bdesgostar\b|\bdesgosto\b|\bdetestar\b|\bempatia\b|\bentusiasmado\b|\bexalt
ado\b|\bfebril\b|\bfebrilmente\b|\bfelizmente\b|\bfrancamente\b|\bgostar\b|\bhomofobia\b|\bhomofóbico\
b|\bimpressionado\b|\binabalado\b|\binfelizmente\b|\binsatisfeito\b|\binteressado\b|\bintimidado\b|\binv
eja\b|\binvejar\b|\birritado\b|\blamentar\b|\blastimar\b|\blastimar\b|\bmedo\b|\bmenosprezar\b|\bnervos
o\b|\bodiar\b|\bódio\b|\bpaciente\b|\bpena\b|\bperturbado\b|\bprantear\b|\bprazer\b|\bpreocupado\b|\br
essentimento\b|\bressentir\b|\bsatisfação\b|\bsatisfeito\b|\bsentir\saversão\b|\bsossegar\b|\bsurtar\b|\b
temer\b|\btomado\b|\btranquilidade\b|\btranquilo\b", str(x))))
  df_frame["Emoção_direcionada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babalado\b|\babatido\b|\babatimento\b|\baborrecido\b|\baborrecimento\b|\badmirado\b|
\baflição\b|\baflito\b|\bafobado\b|\bagitado\b|\bagonia\b|\bagonizado\b|\balarmado\b|\balegria\b|\bamar
gura\b|\bamargurado\b|\bambicioso\b|\bamedrontado\b|\bamor\b|\bangústia\b|\bangustiado\b|\bansios
o\b|\bantipático\b|\barara\b|\bassutado\b|\batordoado\b|\batormentado\b|\bbem\b|\bbravo\b|\bchateaçã
o\b|\bchateado\b|\bchocado\b|\bcondoído\b|\bcontente\b|\bcordialidade\b|\bcurioso\b|\bdecadente\b|\b
decepcionante\b|\bdeleite\b|\bdemolido\b|\bdepressivo\b|\bdesagradável\b|\bdesagrado\b|\bdesanima
do\b|\bdesânimo\b|\bdesapontado\b|\bdesapontamento\b|\bdesconcertado\b|\bdesconfiança\b|\bdesc
```

```
onforto\b|\bdesconsolado\b|\bdescontentamento\b|\bdescontraído\b|\bdesencorajado\b|\bdesencoraja
mento\b|\bdesespero\b|\bdesgastante\b|\bdesgosto\b|\bdesgostoso\b|\bdesolado\b|\bdesorientação\b|\
bdesorientado\b|\bdevastado\b|\bdiversão\b|\bdoloroso\b|\bdor\b|\bembaraçado\b|\bembaraço\b|\bem
ocionado\b|\bempolgado\b|\bencantado\b|\benfurecido\b|\benjoado\b|\bentediado\b|\bentretido\b|\bent
ristecido\b|\benvergonhado\b|\besmagado\b|\bespanto\b|\bestressado\b|\bestupefação\b|\bestupefato\
b|\beuforia\b|\beufórico\b|\bexasperação\b|\bexasperado\b|\bexausto\b|\bexcitação\b|\bexcitado\b|\bex
tasiado\b|\bfarto\b|\bfascinado\b|\bfelicidade\b|\bfeliz\b|\bferido\b|\bfúria\b|\bfurioso\b|\bgraça\b|\bgratifi
cação\b|\bhorror\b|\bhorrorizado\b|\bhumilhação\b|\bhumilhado\b|\binconsolável\b|\bindignado\b|\binqu
ietação\b|\binquieto\b|\binsípido\b|\binteressar\b|\binteresse\b|\birado\b|\birritação\b|\birritado\b|\bjubilo
so\b|\blívido\b|\blúgrube\b|\bluto\b|\bmaravilhado\b|\bmau\b|\bmelancólico\b|\bmiserável\b|\bmistificad
o\b|\bnervoso\b|\bofendido\b|\bofensa\b|\bperplexidade\b|\bperplexo\b|\bperturbado\b|\bpetrificado\b|\
bpreocupação\b|\bpreocupado\b|\bradiante\b|\braiva\b|\brelaxado\b|\brepulsa\b|\bressentido\b|\brevolt
ado\b|\bsaqueado\b|\bsatisfação\b|\bsatisfeito\b|\bsimpatia\b|\bsimpático\b|\bsimpatizar\b|\bsofrimento
\b|\bsombrio\b|\bsurpreendido\b|\bsurpreso\b|\btranstornado\b|\btraumatizado\b|\btriste\b|\btristemente
\b|\btristeza\b|\bvexação\b|\bzangado\b", str(x)))
  df_frame["Emoções_de_atividade_mental"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdesfrutar\b|\bdistrair\b|\bdivertir\b", str(x))))
  df_frame["Emoções_por_estímulo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balegria\b|\bconfundir\b|\bdesanimado\b|\bdeslumbrar\b|\bintrigado\b|\bpreocupar\b|\bs
urpreender\b", str(x))))
  df_frame["Empregar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdespedido\b|\bempregado\b", str(x))))
  df_frame["Encontrar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdar\sde\scara\b", str(x))))
  df_frame["Encontro_hostil"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbatalha\b|\bbriga\b|\bbrigaiada\b|\bbrigar\b|\bconflito\b|\bconfronto\b|\bdesentendimen
to\b|\bdiscussão\b|\bdisputa\b|\bguerra\b|\binsultar\b|\bluta\b|\blutar\b|\bmorder\b|\btiro\b|\btumultuar\b|
\bxingar\b", str(x))))
  df_frame["Enfatizar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bênfase\b|\bfocar\b|\bfoco\b|\bprestar\b", str(x))))
  df_frame["Enterrar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\benterrado\b|\benterrar\b",
str(x))))
  df_frame["Entidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balgo\b|\bcama\b|\bcobertor\b|\bcoisa\b|\bdeus\b|\bentidade\b|\bfigura\b|\bfogão\sà\sle
nha\b|\bfogão\b|\bindivíduo\b|\bitem\b|\blápis\b|\blençol\b|\bmaterial\b|\bmonstro\b|\bnada\b|\bobjeto\b|
\bsofá\b|\btirolesa\b|\btravesseiro\b|\bvasilha\b|\bvela\b", str(x))))
  df_frame["Entidade_biológica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbacilo\b|\bbactéria\b|\bcoco\b|\bcogumelo\b|\bespirilo\b|\bforma\sde\svida\b|\bhumano
\b|\blivre\b|\bmicrorganismo\b|\bmofo\b|\borganismo\b|\bparasita\b|\bprocariota\b|\bunicelular\b|\bvida\
b", str(x))))
  df_frame["Entidade_física"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bátomo\b|\bburaco\snegro\b|\bestelar\b|\bestrela\b|\bpartícula\b|\bsol\b", str(x))))
  df_frame["Entregar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bentrega\b|\bentregar\b",
str(x))))
  df_frame["Entretenimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bclube\b|\bentretenimento\b|\bentreter\b", str(x))))
  df_frame["Envelhecimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamadurecer\b|\bcrescer\b|\benvelhecer\b|\benvelheimento\b", str(x))))
  df_frame["Enviar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bendereçar\b|\benviar\b|\bmandado\b|\bmandar\b", str(x))))
  df_frame["Equipamentos_esportivos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparelho\b|\bapito\b|\barco\b|\bargola\b|\barma\b|\bbandeira\b|\bbarco\b|\bbarra\sfixa\
b|\bbarra\b|\bbarras\sassimétricas\b|\bbarras\sparalelas\b|\bbastão\b|\bbicicleta\b|\bbike\b|\bbola\b|\bc
aiaque\b|\bcâmera\sdigital\b|\bcaneleira\b|\bcanoa\b|\bcapacete\b|\bcartão\b|\bcavalo\scom\salças\b|\
bcavalo\b|\bclipe\snasal\b|\bcoquilha\b|\bcorda\b|\bcotoveleira\b|\bdardo\b|\bdisco\b|\bembarcação\b|\
bequipamento\b|\bespada\b|\bfita\b|\bflecha\b|\bflorete\b|\bjoelheira\b|\bmaça\b|\bmartelo\b|\bmáscara
\sfacial\b|\bmáscara\b|\bmesa\sde\ssalto\b|\bmesa\b|\bóculos\sde\snatação\b|\bóculos\b|\bpena\b|\bp
eso\b|\bpeteca\b|\bpistola\sde\spartida\b|\bpistola\b|\bprancha\b|\bprotetor\sbucal\b|\bprotetor\sde\sca
beça\b|\bprotetor\sde\sgarganta\b|\bprotetor\sde\sorelha\b|\bprotetor\sde\souvido\b|\bprotetor\snasal\b
|\braquete\b|\bremo\b|\bsabre\b|\bskate\b|\bsolo\b|\btaco\b|\btrampolim\b|\btrave\b|\bvara\b|\bvela\b|\b
volante\b", str(x))))
```

```
  df_frame["Escapar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bfugir\b", str(x))))
  df_frame["Escolher"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdiscotecagem\b|\beleger\b|\beleição\b|\bescolha\b|\bescolher\b|\boptar\b|\bselecionar\
b|\bvotação\b|\bvotar\b", str(x))))
  df_frame["Esconder_objetos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\besconder\b|\bescondido\b", str(x))))
  df_frame["Escrutínio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\banalisar\b|\banálise\b|\banalista\b|\banalítico\b|\bbusca\b|\bbuscar\b|\bchecar\b|\benc
arar\b|\bescanear\b|\bescrutinar\b|\bescrutínio\b|\bestudar\b|\bestudo\b|\bexaminar\b|\bexplorado\b|\b
explorar\b|\bfolhear\b|\binspeção\b|\binspecionar\b|\binspetor\b|\bintrometer-
se\b|\binvestigação\b|\binvestigar\b|\bmonitoração\b|\bmonitorar\b|\bnão\smonitorado\b|\bobservar\b|\
bpeneirar\b|\bprocura\b|\bprocurar\b|\breconhecer\b|\breconhecimento\b|\brevisar\b|\brevistar\b|\bson
dar\b|\bvarredura\b|\bvarrer\b|\bvasculhar\b|\bver\b|\bverificar\b|\bvigilância\b", str(x))))
  df_frame["Especialidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badepto\b|\badepto\b|\bamador\b|\bamador\b|\bás\b|\bás\b|\bbem\sversado\b|\bbom\b|
\bcompetência\b|\bcompetente\b|\bconhecedor\b|\bcraque\b|\bdesqualificado\b|\bespecialista\b|\besp
ecializado\b|\besplêndido\b|\bestupêndo\b|\bexcelente\b|\bexperiência\b|\bexperiente\b|\bexpert\b|\bfã
\b|\bfamiliar\b|\bfantástico\b|\bforte\b|\bfraco\b|\bguru\b|\bhabilidade\b|\bhabilidoso\b|\bhorrível\b|\bign
orante\b|\binacreditável\b|\bincompetência\b|\bincompetente\b|\binépcia\b|\binepto\b|\binexperiente\b|\
bleigo\b|\bmaestria\b|\bmago\b|\bmaravilhoso\b|\bmediano\b|\bmedíocre\b|\bmestre\b|\bmestre\b|\bnot
ável\b|\bnovato\b|\bnovo\b|\bótimo\b|\bpró\b|\bproeza\b|\bproficiência\b|\bproficiente\b|\bruim\b|\bsobe
rbo\b|\bsobressair\b|\bsuperlativo\b|\btécnica\b|\bterrível\b|\btremendo\b|\bversado\b|\bvirtuosidade\b|\
bvirtuoso\b", str(x))))
  df_frame["Especificação_individual"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bespecífico\b", str(x))))
  df_frame["Esperar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baguardar\b|\besperar\b",
str(x))))
  df_frame["Esportes"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batletismo\b|\bbadminton\b|\bbaseball\b|\bbasquete\b|\bbasquetebol\b|\bbeisebol\b|\bb
oxe\b|\bcanoagem\b|\bcaratê\b|\bciclismo\b|\bcrossfit\b|\bescalada\b|\besgrima\b|\besporte\b|\besporti
vo\b|\besqueite\b|\besqueitismo\b|\bfutebol\b|\bfutsal\b|\bginástica\b|\bgolfe\b|\bhalterofilismo\b|\bhand
ebol\b|\bhipismo\b|\bhóquei\ssobre\sgrama\b|\bjudô\b|\bkaratê\b|\blevantamento\sde\speso\b|\bluta\so
límpica\b|\bnado\ssincronizado\b|\bnatação\b|\bpentatlo\smoderno\b|\bpólo\saquático\b|\bremo\b|\bru
gbi\b|\brugby\b|\bsalto\sornamental\b|\bskate\b|\bsoftball\b|\bsoftbol\b|\bsurfe\b|\btaekwondo\b|\btênis\
sde\smesa\b|\btênis\b|\btiro\scom\sarco\b|\btiro\sesportivo\b|\btriatlo\b|\bvela\b|\bvôlei\sde\spraia\b|\bv
ôlei\b|\bvoleibol\b", str(x))))
  df_frame["Estado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bestar\b", str(x))))
  df_frame["Estado_continuar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdeixar\b|\bdescansar\b|\bestar\b|\bficar\b|\bmanter\b|\bpermanecer\b|\bprevalecer\b",
str(x))))
  df_frame["Estado_da_entidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomplexo\b|\bcondição\b|\bestado\sde\schoque\b|\bestado\sde\sconsciência\b|\besta
do\b|\bestar\b", str(x))))
  df_frame["Estágio_de_progresso"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balta\stecnologia\b|\bantigo\b|\bavançado\b|\bbaixa\stecnologia\b|\bcontemporâneo\b|\
bde\sponta\b|\bde\súltima\sgeração\b|\bdesenvolvido\b|\bgeração\b|\bmaduro\b|\bmaturidade\b|\bmod
ernizar\b|\bmoderno\b|\bpróxima\sgeração\b|\bsofisticação\b|\bsofisticado\b|\búltima\sgeração\b",
str(x))))
  df_frame["Estar_anexado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\belo\b|\bligado\b|\bponto\sde\sintegração\b|\bsolto\b", str(x))))
  df_frame["Estar_de_acordo_sobre_a_avaliação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconcordar\b", str(x))))
  df_frame["Estar_em_cativeiro"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bpreso\b",
str(x))))
  df_frame["Estar_em_risco"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bsegurança\b|\bseguro\b", str(x))))
  df_frame["Estar_em_vigor"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bválido\b", str(x))))
  df_frame["Estar_molhado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmolhado\b", str(x))))
  df_frame["Estar_no_controle"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badministrado\b|\badministrar\b|\bconseguir\b|\bcontrolar\b", str(x))))
```

```
  df_frame["Estar_separado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdesligamento\b",
str(x))))
  df_frame["Estética"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamável\b|\bbarroco\b|\bbeleza\b|\bbelo\b|\bbonito\b|\bbucólico\b|\belegante\b|\besport
ivo\b|\besteticamente\b|\bestiloso\b|\bfeio\b|\bformosura\b|\bfrescura\b|\bhorrendo\b|\blindo\b|\bpitores
co\b|\bplástico\b|\brequintar\b|\brequinte\b|\brústico\b|\bsaboroso\b", str(x))))
  df_frame["Estimar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badivinha\b|\badivinhar\b|\bestimativa\b", str(x))))
  df_frame["Estimular_emoção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bencantar\b|\birritar\b|\bprazeroso\b|\bsurpresa\b", str(x))))
  df_frame["Estragar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bequivocar\b|\berrar\b",
str(x))))
  df_frame["Estudar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcursar\b|\bdiplomar\b|\bestudar\b", str(x))))
  df_frame["Esvaziar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\benxaguar\b|\besponja\b|\blavável\b|\blimpar\b|\blimpeza\b|\bpolido\b|\bsujeira\b",
str(x))))
  df_frame["Evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacontecer\b|\bacontecimento\b|\bassolar\b|\bcongresso\b|\bdesenvolvimento\b|\bepis
ódio\b|\bevento\b|\bfato\b|\bincidente\b|\bjogo\b|\blotar\b|\bmissa\b|\bocorrer\b|\bprosseguir\b|\bquadro
\b|\brealizado\b|\brealizar\b|\bretiro\b|\bser\b|\bshow\b|\bsituação\b|\bsuceder\b", str(x))))
  df_frame["Evento_desejável"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bboa\sideia\b|\bdever\b|\bmá\sideia\b|\bpoder\b", str(x))))
  df_frame["Evento_esportivo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentação\b|\bbrasileirão\b|\bcombate\b|\bCopa\sAmérica\b|\bCopa\sdo\sMundo\b
|\bCopa\b|\bcorrida\b|\bduelo\b|\bevento\b|\bgame\b|\bjogo\sde\sida\b|\bjogo\sde\svolta\b|\bjogo\sem\
scasa\b|\bjogo\sfora\sde\scasa\b|\bjogo\b|\bjogos\solímpicos\b|\bluta\b|\bmundial\b|\bolimpíada\b|\bolí
mpico\b|\bparaolimpíada\b|\bpartida\sde\sida\b|\bpartida\sde\svolta\b|\bpartida\sem\scasa\b|\bpartida\
sfora\sde\scasa\b|\bpartida\b|\bprova\b|\bregata\b|\btemporada\b", str(x))))
  df_frame["Evento_histórico"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmomento\b",
str(x))))
  df_frame["Evento_social"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbadalar\b|\bbaile\b|\bbalada\b|\bbanquete\b|\bceia\b|\bcelebração\b|\bcelebrar\b|\bch
urrasco\b|\bcomemoração\b|\bconselho\b|\bencontro\b|\bfeira\b|\bfesta\sbeneficente\b|\bfesta\b|\bfest
ejar\b|\bfestival\b|\bhappy\shour\b|\bjantar\b|\bnoitada\b|\bpiquenique\b|\bpromover\b|\brave\b|\brecep
ção\b|\breunião\b|\bsamba\b|\bsocial\b|\bvelório\b", str(x))))
  df_frame["Evidências"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbasear\b|\bevidência\b|\bevidenciar\b|\bindicar\b|\bindicativo\b|\bindício\b|\bmanifesta
r\b|\bpostulado\b|\bprova\b", str(x))))
  df_frame["Evitar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bevitar\b", str(x))))
  df_frame["Exatidão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcerto\b|\bcorreto\b|\bcorrigir\b|\bdireito\b|\berrar\b|\bexato\b|\bpreciso\b", str(x))))
  df_frame["Exemplar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmodelo\b|\bpadrão\b|\bparadigma\b", str(x))))
  df_frame["Exercitar-se"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bexercício\b", str(x))))
  df_frame["Existência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babstrato\b|\bconcreto\b|\bencontrar\b|\bestar\b|\bexistência\b|\bexistente\b|\bexistir\b|\
bhaver\b|\bpermanecer\b|\breal\b|\brealidade\b|\bter\b", str(x))))
  df_frame["Existência_circunscrita"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bsurgir\b",
str(x))))
  df_frame["Expansão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcrescer\b|\bexpansão\b|\bextensão\b|\binflação\b", str(x))))
  df_frame["Expectativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdever\b|\bespera\b|\besperar\b|\bexpectativa\b|\bimprevisibilidade\b|\bsonhar\b",
str(x))))
  df_frame["Expectativa_classificada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapenas\b|\bdimensão\b|\bgeral\b|\binteiro\b|\bmero\b|\btamanho\b|\btodo\b", str(x))))
  df_frame["Experenciar_ferimento_corporal"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrebentar\b|\bferimento\b|\bfurar\b|\bmachucar\b|\bquebrar\b|\bsangrar\b|\btorcer\b",
str(x))))
```

```
  df_frame["Experiência_de_percepção"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcheirar\b|\bcompreender\b|\bdelirar\b|\bdelírio\b|\bdetectar\b|\bescutar\b|\bexperiência
\b|\bexperimentar\b|\binvisível\b|\bouvir\b|\bperceber\b|\bpercepção\b|\bpesadelo\b|\bsaborear\b|\bsen
tir\b|\bsonhar\b|\bsonho\b|\btestemunhar\b|\bver\b|\bvivenciar\b", str(x)))
  df_frame["Experimentação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btratamento\b",
str(x))))
  df_frame["Experimentar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexperimentar\b|\bvivenciar\b", str(x))))
  df_frame["Expressão_facial"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcareta\b|\bsorriso\b", str(x))))
  df_frame["Expressar_publicamente"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexpressar\b|\bmanifestar\b|\bpassar\b|\bvoz\b", str(x))))
  df_frame["Extensão_linear_de_medidas"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bano-
luz\b|\bjarda\b|\bkm\b|\bmetro\b|\bmilha\b|\bmilímetro\b|\bpolegada\b|\bquilômetro\b", str(x))))
  df_frame["Fama"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcelebridade\b|\bconhecido\b|\bépico\b|\bestatura\b|\bfama\b|\bfamoso\b|\bfamoso\b|\b
fazer\snome\spara\salguém\b|\bgrande\snome\b|\binfame\b|\blendário\b|\bnotoriedade\b|\bnotório\b|\b
ovelha\snegra\b|\brenomado\b|\brenome\b|\breputação\b", str(x))))
  df_frame["Familiaridade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconhecer\b|\bconhecido\b|\bdesconhecido\b|\bfamiliar\b|\bintimista\b|\bíntimo\b|\bnov
o\b", str(x))))
  df_frame["Fase_preliminar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bclassificação\b|\bconquistar\svaga\b|\beliminatórias\b|\bfase\sclassificatória\b|\bfase\s
de\sgrupos\b|\bfase\spreliminar\b|\bgrupo\b|\bpreliminares\b|\bvaga\b", str(x))))
  df_frame["Fazedores_de_barulho"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batabaque\b|\bcandongueiro\b|\bchocalho\b|\bgaita\sde\sfole\b|\btambor\b|\btrombeta\
b", str(x))))
  df_frame["Fazer_barulho"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balgazarra\b|\bbarulho\b|\bcanto\b|\bchorar\b|\bgargalhada\b|\bgritar\b|\bguincho\b|\bre
ssoar\b|\brir\b|\bsoluçar\b|\btrovejar\b|\bzoar\b", str(x))))
  df_frame["Fazer_câmbio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcâmbio\b|\btroca\b|\btrocar\b", str(x))))
  df_frame["Fazer_compras"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcompras\b", str(x))))
  df_frame["Fazer_turismo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacampar\b|\bapreciar\b|\baproveitar\b|\bconhecer\b|\bcurtir\b|\bdesfrutar\b|\bfazer\stur
ismo\b|\bpaisagem\b|\breceber\b|\btour\b|\bturismo\sferroviário\b|\bturismo\sgastronômico\b|\bturismo\
b|\bturístico\b|\bver\b|\bvisita\b|\bvisitação\b|\bvisitar\b|\bvista\b|\bvisual\b", str(x))))
  df_frame["Fechamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babrir\b|\bfechar\b|\btampar\b", str(x))))
  df_frame["Fechamento_de_locais"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bfechar\b",
str(x))))
  df_frame["Fenômenos_naturais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamanhecer\b|\bamanhecer\b", str(x))))
  df_frame["Final"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconquistar\b|\bentregar\b|\bfinal\b|\bganhar\b|\bperder\b|\btítulo\b|\bvencer\b", str(x))))
  df_frame["Finalidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balvo\b|\bde\smodo\sa\b|\bde\b|\bdeterminado\b|\bfinalidade\b|\bintenção\b|\bintuito\b|\
bmotivo\b|\bobjetivo\b|\bobjeto\b|\bpara\sque\b|\bpara\b|\bplanejar\b|\bplano\b|\bpra\b|\bpretender\b|\b
pretendido\b|\bpropósito\b|\bproposta\b|\bresolvido\b|\broteiro\b|\buso\b|\bvisar\b", str(x))))
  df_frame["Finalidade_do_utensílio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfunção\b|\brecomendar\b|\buso\b", str(x))))
  df_frame["Financiamento"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bfundar\b", str(x))))
  df_frame["Fingir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcavar\b|\bfingir\b|\bsimular\b",
str(x))))
  df_frame["Foco_de_estímulo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babominável\b|\baconchegante\b|\bagonizante\b|\bagradável\b|\bagravação\b|\bagrava
nte\b|\balarmante\b|\balucinante\b|\bameno\b|\bangustiante\b|\banimado\b|\banimador\b|\bapaixonant
e\b|\bapaziguador\b|\bapetitoso\b|\baprazível\b|\bapreciável\b|\bapresentável\b|\barrebatador\b|\barre
piante\b|\barrepio\b|\bassustador\b|\baterrorizante\b|\batormentador\b|\bbacana\b|\bbem-
humorado\b|\bcalmante\b|\bcansativo\b|\bcativante\b|\bcharmoso\b|\bchato\b|\bcheio\b|\bchocante\b|\
```

```
bcômico\b|\bcomodidade\b|\bcomovente\b|\bconfortante\b|\bconfortável\b|\bconfuso\b|\bconsolador\b|\
bconstrangedor\b|\bdelícia\b|\bdelicioso\b|\bdepressivo\b|\bdesagradável\b|\bdesapontador\b|\bdesbar
atado\b|\bdescanso\b|\bdesconcertante\b|\bdesconfortável\b|\bdesencorajador\b|\bdesmotivante\b|\bd
esorientante\b|\bdevastador\b|\bdivertido\b|\beletrizante\b|\bemocionante\b|\bempolgante\b|\bencanta
dor\b|\bencorajador\b|\benfadonho\b|\benfurecedor\b|\bengraçado\b|\benlouquecedor\b|\bentristecedo
r\b|\benvolvente\b|\bespantoso\b|\bestimulante\b|\bestremecedor\b|\bestressante\b|\bestupeficante\b|\
bexasperador\b|\bfascinante\b|\bformidável\b|\bfrio\b|\bglamour\b|\bgostoso\b|\bgratificante\b|\bhilário\
b|\bhumilhante\b|\bimpressionante\b|\bincitador\b|\bincômodo\b|\bincrível\b|\binquietante\b|\binsatisfat
ório\b|\binsultante\b|\bintimidador\b|\bintrigante\b|\birritação\b|\birritante\b|\birritante\b|\blamentável\b|\
blegal\b|\bmarcante\b|\bmistificante\b|\bmonótono\b|\bmortificante\b|\bnojeira\b|\bnojento\b|\bofensivo\
b|\bpacificador\b|\bpatético\b|\bperturbador\b|\bperturbar\b|\bpreocupante\b|\bproblemático\b|\bproveit
o\b|\bquerido\b|\brecreação\b|\brelaxamento\b|\brelaxante\b|\brelaxar\b|\brepugnante\b|\brepulsivo\b|\
brevigorante\b|\brevoltante\b|\brico\b|\bsatisfatório\b|\bsério\b|\bsinistro\b|\bsolene\b|\bsuculento\b|\bs
urpreendente\b|\bsuspense\b|\btedioso\b|\bterrível\b|\btocante\b|\btranquilizador\b|\btraumático\b|\btra
umatizante\b|\btriste\b|\bvazio\b", str(x)))
  df_frame["Formar_relações"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bazaração\b|\bboda\b|\bcasar\b|\bnamorar\b|\bperto\b|\bseparar\b|\bunir\b", str(x))))
  df_frame["Formas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bforma\b|\bformar\b|\binclinado\b|\bíngreme\b|\blinha\b|\bperfil\b|\bredondo\b", str(x))))
  df_frame["Fornecimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfornecer\b|\bproporcionar\b|\bservido\b|\bservir\b", str(x))))
  df_frame["Fracasso_de_empreendimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfalir\b", str(x))))
  df_frame["Frequência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagora\b|\banual\b|\banualmente\b|\bàs\svezes\b|\bbianual\b|\bbimestral\b|\bcomum\b|
\bconstantemente\b|\bcotidiano\b|\bcotidiano\b|\bde\stempos\sem\stempos\b|\bde\svez\sem\squando\
b|\bdesta\svez\b|\bdiariamente\b|\bdiário\b|\bdiário\b|\besporádico\b|\bfrequência\b|\bfrequente\b|\bfre
quentemente\b|\bgeralmente\b|\binfrequente\b|\binfrequentemente\b|\bintermintente\b|\bintervalo\b|\b
mensalmente\b|\bnormal\b|\bnormalmente\b|\bnoturno\b|\bnunca\smais\b|\bnunca\b|\bo\stempo\stodo\
b|\bocasional\b|\bocasionalmente\b|\bordináriamente\b|\bperiódico\b|\bperíodo\b|\bquinzenalmente\b|\
bquotidiano\b|\bquotidiano\b|\braramente\b|\braro\b|\brecorrência\b|\brecorrente\b|\bregular\b|\bregula
rmente\b|\brepetir\b|\bsemanalmente\b|\bsemestre\b|\bsempre\b|\bsomente\b", str(x)))
  df_frame["Frugalidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdesperdiçar\b", str(x))))
  df_frame["Função"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bservir\b", str(x))))
  df_frame["Ganhar_um_prêmio"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvitória\b",
str(x))))
  df_frame["Ganhos_e_perdas"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcusto-
benefício\b|\bganhar\b|\brender\b", str(x))))
  df_frame["Grau"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babsolutamente\b|\bdeveras\b|\bem\sparte\b|\benorme\b|\bestupidamente\b|\bextrema
mente\b|\bextremo\b|\bgrande\b|\bligeiramente\b|\bmais\b|\bmenos\b|\bmuito\b|\brealmente\b|\btanto\
b|\btão\b|\btotalmente\b", str(x))))
  df_frame["História"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bhistória\b", str(x))))
  df_frame["Hospedar-se"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bficar\b|\bhospedar\b",
str(x))))
  df_frame["Hospital"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bhospital\b", str(x))))
  df_frame["Hospitalidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bacolhedor\b", str(x))))
  df_frame["Idade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badolescência\b|\badulto\b|\bantigo\b|\bcom\b|\bde\b|\bidade\b|\binfância\b|\binfantil\b|\
bjovem\b|\bmaduro\b|\bmeninice\b|\bnovo\b|\bter\b|\bvelhice\b|\bvelho\b", str(x))))
  df_frame["Identidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bidentidade\b", str(x))))
  df_frame["Idiossincrasia"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpeculiar\b|\bpessoal\b|\bprivativo\b|\búnico\b", str(x))))
  df_frame["Impacto"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcolidir\b|\bpaulada\b|\bporrada\b", str(x))))
  df_frame["Impedir_ou_permitir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baprovar\b|\bdar-
se\sao\sluxo\b|\bdeixar\b|\binadimissível\b|\binviabilizar\b|\bpermitir\b", str(x))))
  df_frame["Importância"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcentral\b|\bconhecido\b|\bcrítico\b|\bdominar\b|\bgravemente\b|\bimperdível\b|\bimpor
```

```
tância\b|\bimportante\b|\bmarco\b|\bprimário\b|\bprincipal\b|\bprivilegiar\b|\bqualificar\b|\bsecundário\b|
\bselo\b|\bsimbolo\b", str(x)))
  df_frame["Impor_obrigação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexigir\b|\bobrigar\b", str(x))))
  df_frame["Impressão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparência\b|\bimagem\b|\bimpressionar\b", str(x))))
  df_frame["Impulso_biológico"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfaminto\b|\bfome\b", str(x))))
  df_frame["Inclinação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvocação\b", str(x))))
  df_frame["Inclusão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babranger\b|\bagrupar\b|\baté\b|\bcom\b|\bcontar\b|\bconter\b|\benglobar\b|\benvolver\
b|\bincluir\b|\bincorporar\b|\bjuntar\b|\bmisturar\b|\bpossuir\b|\breunir\b", str(x))))
  df_frame["Incremento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balém\sde\b|\bmais\b|\boutro\b|\bsomar\b", str(x))))
  df_frame["Indicar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bacusar\b", str(x))))
  df_frame["Inefabilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmagia\b|\bmágica\b|\bmágico\b", str(x))))
  df_frame["Influência_objetiva"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafetar\b|\befeito\b|\bimpactar\b|\bimpacto\b|\binfluência\b|\binfluenciar\b|\bpoder\b|\bp
rejudicar\b|\bprejuízo\b|\bprocurar\b", str(x))))
  df_frame["Influência_subjetiva"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconvidativo\b|\bdesestimular\b|\bembriagado\b|\binfluenciar\b|\binspiração\b|\binspira
dor\b|\binspirar\b|\bmusa\b|\btrazer\b|\bvaler\sa\spena\b", str(x))))
  df_frame["Informação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdado\b|\bdados\b|\bdica\b|\binformação\b|\binformar\b|\bnoticiar\b", str(x))))
  df_frame["Informação_atribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bde\sacordo\scom\b|\bsegundo\b", str(x))))
  df_frame["Informação_não_atribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bsupostamente\b", str(x))))
  df_frame["Infrações"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfalha\b|\bfalta\b|\binfração\b|\bmarcar\sfalta\b", str(x))))
  df_frame["Infrações_diretas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcarrinho\b|\bderrubar\b|\bentrada\b|\bsplashing\b", str(x))))
  df_frame["Infrações_indiretas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcarregada\b|\bcarregar\b|\bcavar\b|\bcondução\b|\bconduzir\b|\bdois\stoques\b|\bdupl
a\sfalta\b|\bfalta\sde\spé\b|\bimpedimento\b|\binvadir\b|\binvasão\b|\bjogo\sperigoso\b|\bmão\b|\bquei
mar\sa\slargada\b|\bqueimar\b|\bsimulação\b|\bsimular\b", str(x))))
  df_frame["Infraestrutura"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbase\b|\binfraestrutura\b", str(x))))
  df_frame["Ingestão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balimentação\b|\balimentar\b|\balmoçar\b|\balmoço\b|\bbeber\b|\bbrocar\b|\bcomer\b|\b
comida\b|\bconsumir\b|\bjantar\b|\blanchar\b|\blanche\b|\bpetiscar\b|\btomar\b", str(x))))
  df_frame["Ingredientes"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babacaxi\b|\baçaí\b|\baçúcar\sde\sconfeiteiro\b|\baçúcar\b|\bágua\b|\baguardente\b|\ba
ipim\b|\bálcool\b|\balho\b|\banchova\b|\barroz\b|\bazeite\sde\sdendê\b|\bazeite\sde\soliva\b|\bazeite\b|
\bbacalhau\b|\bbacon\b|\bbacuri\b|\bbadejo\b|\bbanana-da-
terra\b|\bbanana\b|\bbanha\sde\sporco\b|\bbatata-
baroa\b|\bbatata\b|\bbife\b|\bburiti\b|\bcação\b|\bcacau\b|\bcachaça\b|\bcafé\b|\bcajá\b|\bcaju\b|\bcald
o\sde\scarne\b|\bcamarão\b|\bcana-de-
açúcar\b|\bcana\b|\bcanela\b|\bcapivara\b|\bcaranguejo\b|\bcarne-
seca\b|\bcarne\b|\bcarneiro\b|\bcatupiry\b|\bcavaquinha\b|\bcebola\b|\bcereal\b|\bcerveja\b|\bchantili\b
|\bchantilly\b|\bcharque\b|\bcheddar\b|\bchocolate\sem\spó\b|\bchocolate\sgranulado\b|\bchocolate\b|\
bchuchu\b|\bcoalhada\b|\bcoco\b|\bcoentro\b|\bcontra-
filé\b|\bcostela\b|\bcravo\b|\bcrustáceo\b|\bcupuaçu\b|\bdendê\b|\bdoce\sde\sleite\b|\berva-
mate\b|\bfarinha\sde\smandioca\b|\bfarinha\b|\bfécula\b|\bfeijão\b|\bfermento\b|\bfilé\b|\bfrango\b|\bfrut
a\b|\bfruto\b|\bgalinha\b|\bgorgonzola\b|\bguaraná\b|\bhortelã\b|\bingrediente\b|\biogurte\b|\bjaca\b|\bj
ambu\b|\bjavali\b|\bjoelho\sde\sporco\b|\bketchup\b|\blagarto\b|\blagosta\b|\blagostim\b|\blegume\b|\bl
eite\scondensado\b|\bleite\b|\blinguiça\scalabresa\b|\blinguiça\b|\blombo\b|\bmacaxeira-
brava\b|\bmacaxeira\b|\bmaionese\b|\bmandioca-
brava\b|\bmandioca\b|\bmanga\b|\bmangaba\b|\bmaniva\b|\bmanteiga\b|\bmarisco\b|\bmassa\b|\bmel\
```

```
b|\bmilho\b|\bmoranga\b|\bmorango\b|\bmozzarela\b|\bmurici\b|\bnoz\b|\bnutella\b|\bóleo\b|\bora-pro-
nóbis\b|\bovo\b|\bpaçoca\b|\bpacu\b|\bpaio\b|\bpão\b|\bpeito\sde\sfrango\b|\bpeixe\b|\bpequi\b|\bpernil
\b|\bperu\b|\bpicanha\b|\bpimenta\b|\bpinhão\b|\bpiranha\b|\bpirarucu\b|\bpolvilho\b|\bqueijo\b|\bquiab
o\b|\bquirera\b|\brepolho\b|\bsal\b|\bsalmão\b|\bsalsicha\b|\bsalsichão\b|\bsapoti\b|\bsardinha\b|\bshita
ke\b|\bsobrecoxa\b|\bsteak\b|\btapioca\b|\btempero\b|\btomate\b|\btorresmo\b|\btortelli\b|\btucumã\b|\b
tucunaré\b|\btucupi\b|\bumbu\b|\bvegetal\b|\bvinho\b|\bwasabi\b|\bwurst\b", str(x))))
  df_frame["Instalações_esportivas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bárea\spara\scanoagem\b|\barena\b|\bcampo\sde\satletismo\b|\bcampo\sde\sbeisebol\
b|\bcampo\sde\sfutebol\b|\bcampo\sde\sgolfe\b|\bcampo\sde\shóquei\b|\bcampo\sde\srúgbi\b|\bcamp
o\spara\sequitação\b|\bcampo\b|\bcentro\saquático\b|\bcentro\sde\sginástica\solímpica\b|\bcircuito\b|\
bestádio\sde\sfutebol\b|\bestádio\b|\bginásio\spoliesportivo\b|\bginásio\b|\binstalação\b|\blagoa\b|\bma
r\b|\bpavilhão\b|\bpista\sde\satletismo\b|\bpista\sde\sciclismo\b|\bpista\b|\bpraia\b|\bquadra\sde\sbad
minton\b|\bquadra\sde\sbasquete\b|\bquadra\sde\shandebol\b|\bquadra\sde\stênis\b|\bquadra\sde\svô
lei\b|\bquadra\b|\brua\b|\bsambódromo\b|\bvelódromo\b", str(x))))
  df_frame["Instância"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcomo\b|\bexemplo\b",
str(x))))
  df_frame["Instância_de_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bciclo\b|\bde\snovo\b|\bfase\b|\bnovamente\b|\bocasião\b|\brepetido\b|\buma\svez\b|\b
vez\b", str(x))))
  df_frame["Instância_única"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcada\b|\bsimplesmente\b|\bsó\b|\búnico\b", str(x))))
  df_frame["Instituições"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\binstituição\b", str(x))))
  df_frame["Intérpretes_e_papéis"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentação\b|\bapresentar\b|\bassistir\b|\batuar\b|\bbrincar\b|\bensaiar\b|\bespetác
ulo\b|\besquete\b|\bestrela\b|\bestrelar\b|\bfazer\b|\bfilme\b|\bpalco\b|\bpapel\b|\bpeça\b|\bprotagoniza
r\b|\bser\b|\bteatro\b|\btreinar\b", str(x))))
  df_frame["Intervenção_médica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentar\b|\bexame\b|\bmedicar\b|\breceitar\b|\bremédio\b|\btraqueotomia\b|\bvítim
a\b", str(x))))
  df_frame["Jogadas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bestilo\b|\bjogada\b|\bjogar\b|\blance\b|\bmanobra\b|\btécnica\b", str(x))))
  df_frame["Jogadas_individuais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacertar\b|\bafundar\b|\bagachamento\b|\bagachar\b|\bagarramento\b|\bagarrar\b|\bali
nhamento\b|\bamorti\b|\bapproach\b|\baproveitar\srebote\b|\baproximação\b|\barranque\b|\barremess
ar\b|\barremesso\b|\batacar\b|\bataque\b|\batirar\b|\bavançar\b|\bavanço\b|\bback\sswing\b|\bbackhan
d\sclear\b|\bbackhand\b|\bbate-
pronto\b|\bbater\b|\bbatida\b|\bbicicleta\b|\bboggey\b|\bborboleta\b|\bbraçada\b|\bcabeceada\b|\bcabe
cear\b|\bcabeceio\b|\bcaminhar\b|\bchina\b|\bchop\b|\bchutar\b|\bchute\b|\bcobrança\b|\bcobrar\b|\bco
ncha\b|\bcorrer\b|\bcorrida\b|\bcortada\b|\bcortar\b|\bcostas\b|\bcrawl\b|\bcrol\b|\bcruzada\b|\bcruzar\b|
\bdeixadinha\b|\bdisparar\b|\bdouble\sboggey\b|\bdrive\b|\bdrop\sgoal\b|\bdrop\sshot\b|\bdrop\b|\beagl
e\b|\bempurrão\b|\berguer\b|\bescalar\b|\bescanteio\b|\bespalmar\b|\bestilo\slivre\b|\bflick\b|\bforehan
d\b|\bfresh\sair\b|\bfuga\b|\bgirar\b|\bgiro\b|\blançamento\b|\blançar\b|\blance\slivre\b|\blance-
livre\b|\blateral\b|\blevantamento\b|\blevantar\b|\blineout\b|\blivre\b|\bmarco\b|\bmedley\b|\bmeio\spas
so\b|\bnadar\b|\bnado\slivre\b|\bnado\b|\bobstrução\b|\bpancada\sleve\b|\bparalela\b|\bpassada\b|\bp
egar\srebote\b|\bpegar\b|\bpeito\b|\bpeixinho\b|\bpênalti\b|\bpenalty\sgoal\b|\bpernada\b|\bpiaffe\b|\bp
ontapé\sde\spenalidade\b|\bpontapé\sde\sressalto\b|\bpontapé\b|\bprogressão\b|\bpular\b|\bpulo\b|\bp
ush\sand\spump\b|\bpush-
hit\b|\bquicar\b|\bquique\b|\brebote\b|\bremada\b|\bremar\b|\brolamento\b|\brolar\b|\bsacar\b|\bsaltar\b
|\bsalto\stesoura\b|\bsalto\b|\bsaque\b|\bsegurar\b|\bserviço\b|\bservir\b|\bshot\b|\bsmash\b|\bsoltar\b|\
bsprint\b|\bswing\b|\btacada\b|\btacar\b|\btesoura\b|\btiro\sde\scanto\b|\btiro\sde\sgol\b|\btiro\sde\smet
a\b|\btiro\slivre\b|\btocar\b|\btopspin\b|\btoque\b|\bv\b|\bvelejar\b|\bvoleio\b", str(x))))
  df_frame["Jogadas_interativas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagarramento\b|\bagarrar\b|\barremessar\b|\barremesso\b|\batacar\b|\bataque\b|\bbate
r\b|\bblock\spass\b|\bblock\b|\bbloquear\b|\bbloqueio\b|\bbola\salta\b|\bcarretilha\b|\bchapéu\b|\bchuta
r\b|\bchute\b|\bclear\b|\bclinche\b|\bcombinar\b|\bcontra-atacar\b|\bcontra-
ataque\b|\bcruzado\b|\bcruzamento\b|\bcruzar\b|\bdefender\b|\bdefesa\b|\bdefletir\b|\bdeflexão\b|\bder
rubada\b|\bderrubar\b|\bdevolução\b|\bdevolver\b|\bdireto\b|\bdriblar\b|\bdrible\sda\svaca\b|\bdrible\b|\
berguer\b|\bescalão\b|\besquiva\b|\besquivar\b|\bestabilização\b|\bestrangulamento\b|\bfinta\b|\bganc
ho\b|\bgolpe\b|\bgolpear\b|\bimobilização\b|\bimobilizar\b|\binterceptação\b|\binterceptar\b|\bjab\b|\bkn
ockdown\b|\blambreta\b|\blançamento\b|\blançar\b|\blençol\b|\bleque\b|\blevantamento\b|\blevantar\b|\
```

```
blivrar\b|\blob\b|\blutar\b|\bmarcação\b|\bmarcar\b|\bmaul\b|\bmeia-
lua\b|\bparada\b|\bpassagem\sdo\sbastão\b|\bpassagem\b|\bpassar\b|\bpasse\smolhado\b|\bpasse\sseco\b|\bpasse\b|\bpontapé\b|\bpressionar\b|\bqueda\b|\breceber\b|\brecepção\b|\broubo\sde\sbola\b|\bruck\b|\bsocar\b|\bsoco\b|\bswing\b|\btabela\b|\btabelar\b|\btackle\b|\btocar\b|\btoco\b|\btoque\b|\btroca\b|\btrocar\b|\bultrapassar\b|\buppercut\b", str(x))))
  df_frame["Jogadas_pontuadas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babertura\b|\babrir\smarcador\b|\babrir\splacar\b|\bace\b|\bacertar\b|\badolph\b|\balbatross\b|\bampliar\b|\barco\b|\barremesso\b|\bassistência\b|\bback\shalf\stwist\b|\bback\sthree\squarter\b|\bback\b|\bbalançar\b|\bbalanço\b|\bball\sout\b|\bbandeja\b|\bbarani\sout\b|\bbarani\b|\bbirdie\b|\bbombeiro\b|\bbreak-
point\b|\bcan\scan\b|\bcarpado\b|\bcesta\b|\bchave\b|\bcody\b|\bcompletar\b|\bcompulsory\b|\bconcluir\b|\bconclusão\b|\bconner\sspin\b|\bconverter\b|\bcravada\b|\bcravar\b|\bcruzado\b|\bdecolagem\b|\bdiamidov\b|\bdireto\b|\bdouble\sback\b|\bdouble\sfull\b|\bdouble\smini\stramp\b|\bduplo\stwist\scarpado\b|\bduplo-
duplo\b|\bempunhaduras\b|\benterrada\b|\benterrar\b|\bequilíbrio\b|\bespacate\b|\bespacato\b|\bespargata\b|\bestabilização\b|\bestendida\b|\besticada\b|\bfinalização\b|\bfinalizar\b|\bflic-
flac\b|\bfliffis\b|\bflutuador\b|\bfront\sfull\b|\bfront\sthree\squarter\b|\bfront\b|\bfull\b|\bgame\spoint\b|\bgirar\b|\bgiro\sgigante\b|\bgiro\b|\bgol\scontra\b|\bgol\solímpico\b|\bgol\b|\bgolden\sscore\b|\bgrupado\b|\bguindaste\b|\bgut\swrench\b|\bhalf\sin\shalf\sout\b|\bhalf\snelson\b|\bhalf\b|\bhandspring\b|\bhole\sin\sone\b|\bhypolito\strês\b|\bin\b|\bippon\b|\bjanz\b|\bkoka\b|\bkorbut\b|\blargada\b|\blargar\b|\bmão\saberta\b|\bmarcar\sgol\b|\bmarcar\b|\bmatch\spoint\b|\bmidle\b|\bmiller\b|\bmoinho\b|\bmortal\b|\bmorte\ssúbita\b|\bnocaute\b|\bonda\b|\bout\b|\bparada\sde\smãos\b|\bparada\b|\bparar\b|\bpegada\b|\bpegar\b|\bpike\b|\bpirueta\b|\bpivot\b|\bpivote\b|\bponte\saérea\b|\bponto\b|\bpontuação\b|\bpullover\b|\brandolph\b|\brandy\b|\bretomada\b|\bretomar\b|\brolê\b|\bround-
off\b|\brudolph\b|\brudy\sout\b|\brudy\b|\bsaída\b|\bsaltar\b|\bsalto\spak\b|\bsalto\b|\bset\spoint\b|\bside\b|\bstützkehre\b|\bsuple\b|\bsuplê\b|\btakedown\b|\btkachev\b|\btocar\b|\btoque\b|\btriffis\b|\btriple\sback\b|\btriplo-
duplo\b|\btuck\b|\bvertical\b|\bvéu\b|\bvoar\b|\bvoluntary\b|\bvoo\b|\bwazari\b|\bwhipback\b|\bwipe-
out\b|\byuko\b", str(x))))
  df_frame["Julgamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badmirar\b|\bapaixonado\b|\bdelicioso\b|\berrar\b|\bestigmatizar\b|\bhonesto\b|\bicônico\b|\bimpecável\b|\brespeito\b|\bvalioso\b|\bvalorizar\b", str(x))))
  df_frame["Julgamento_de_intensidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bforte\b", str(x))))
  df_frame["Legalidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcertificado\b|\bdireito\b|\blegal\b", str(x))))
  df_frame["Lembrar_experiência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\besquecer\b|\binesquecível\b|\blembrança\b|\blembrar\b|\bmemória\b", str(x))))
  df_frame["Ler"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bleitor\b|\bleitura\b|\bler\b",
str(x))))
  df_frame["Levar_tempo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bágil\b|\bdevagar\b|\bdevagarzinho\b|\bem\b|\bgradualmente\b|\blentamente\b|\blento\b|\blevar\b|\bligeiramente\b|\bprestatividade\b|\bpresteza\b|\brapidamente\b|\brápido\b", str(x))))
  df_frame["Level_of_force_exertion"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bforça\b|\bforte\b|\bimpotente\b|\bpoderoso\b|\bpotência\b|\bsuave\b", str(x))))
  df_frame["Level_of_force_resistance"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdureza\b|\bduro\b|\belástico\b|\bmais\b|\bresistente\b|\bsensível\b", str(x))))
  df_frame["Libertar_prisioneiro"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\blibertar\b",
str(x))))
  df_frame["Licença_temportária"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bférias\b",
str(x))))
  df_frame["Liderança"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badministrar\b|\bcapitão\b|\bgovernante\b|\blíder\b|\bliderado\b|\bprincesa\b|\breger\b|\brei\b", str(x))))
  df_frame["Limitação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bbarreira\b", str(x))))
  df_frame["Limiting"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapenas\b|\blimitação\b|\blimitar\b|\bsó\b", str(x))))
  df_frame["Locais_naturais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bar\slivre\b|\barquipélago\b|\barrecife\b|\bbacia\b|\bbaía\b|\bbalnear\b|\bbalneário\b|\bbarra\b|\bbarragem\b|\bbeleza\snatural\b|\bbosque\b|\bcachoeira\b|\bcampo\b|\bcanal\b|\bcascata\b|\bc
```

```
atarata\b|\bcaudal\b|\bcaverna\b|\bchapada\b|\bcordilheira\b|\bcórrego\b|\bdeserto\b|\bduna\b|\bensea
da\b|\bestreito\b|\bfloresta\b|\bgaláxia\b|\bgruta\b|\bhidrotermal\b|\bilha\b|\bjardim\b|\blago\b|\blagoa\b|
\blençol\b|\bmangue\b|\bmar\b|\bmargem\b|\bmata\b|\bmirante\b|\bmontanha\b|\bmontão\b|\bmonte\b|
\bmorro\b|\bmundo\b|\bnatural\b|\bnatureza\b|\boceano\b|\borla\b|\bpantanal\b|\bparadisíaco\b|\bparq
ue\secológico\b|\bparque\smunicipal\b|\bparque\snacional\b|\bparque\b|\bpasto\b|\bpenínsula\b|\bpico
\sde\smontanha\b|\bpiscina\snatural\b|\bponto\spanorâmico\b|\bpraia\b|\bqueda\sd\ságua\b|\bqueda\s
de\ságua\b|\brecife\b|\breserva\snatural\b|\breserva\b|\briacho\b|\bribeira\b|\bribeirão\b|\brio\b|\bsertão
\b|\btrilha\sde\scaminhada\b|\bvale\b", str(x))))
  df_frame["Locais_políticos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baldeia\b|\barquidiocese\b|\bbairro\b|\bcapital\b|\bcidade\b|\bcongresso\b|\bcontinente\
b|\bdiocese\b|\bdistrito\b|\bestado\b|\beuropa\b|\bexterior\b|\bfavela\b|\bgoverno\b|\binternacionalment
e\b|\bmetrópole\b|\bmundo\b|\bmunicipal\b|\bmunicípio\b|\bpaís\b|\bparóquia\b|\bplanalto\b|\bpovoado
\b|\bprincipado\b|\btaba\b|\bterra\b|\bvila\b|\bvilarejo\b", str(x))))
  df_frame["Locais_por_colocação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\blocalização\b|\bposição\b", str(x))))
  df_frame["Locais_por_entidade_características"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbairro\b|\bcinturão\b|\benclave\b", str(x))))
  df_frame["Locais_por_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcampo\sde\sbatalha\b|\bcampo\b|\bcena\b|\bcenário\b|\bespaço\b|\blocal\b|\bpicadeir
o\b|\bteatro \b", str(x))))
  df_frame["Locais_por_propriedade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpropriedade\b|\bterreno\b", str(x))))
  df_frame["Locais_por_uso"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bárea\sde\srecreação\b|\bárea\sindustrial\b|\bassociação\ssocial\b|\bassociação\b|\bb
ar\b|\bcadeia\b|\bcanto\sdo\ssilêncio\b|\bcárcere\b|\bcasa\sde\sshow\b|\bcemitério\b|\bcentro\seducac
ional\b|\bchafariz\b|\bcidade\sbase\b|\bcidade\ssede\b|\bcomplexo\b|\bescola\sde\sartes\b|\bescola\sd
e\sbalé\b|\bescola\sde\smúsica\b|\bescola\stécnica\b|\bescola\b|\bfaculdade\sde\sdireito\b|\bfaculdad
e\sde\sodontologia\b|\bfaculdade\b|\bfazenda\b|\bfundação\b|\bigreja\b|\bindústria\b|\binstituição\sde\s
ensino\b|\binstituição\seducacional\b|\binstituição\b|\binterior\b|\bmarco\shistórico\b|\bmeca\b|\bmonu
mento\b|\borganização\sde\sproteção\sdos\sanimais\b|\borganização\sde\sserviço\ssocial\b|\borganiz
ação\ssem\sfins\slucrativos\b|\borganização\b|\bporto\b|\bpraça\b|\bprisão\b|\bpub\b|\bquarto\b|\bquint
al\b|\bsantuário\b|\bsede\b|\bseminário\b|\bsindicato\b|\buniversidade\sparticular\b|\buniversidade\b|\b
UTI\b", str(x))))
  df_frame["Local"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bambiente\b|\bárea\b|\bcentral\b|\bcentro\sda\scidade\b|\bcentro\b|\bespaço\b|\blocal\b
|\blocalidade\b|\blocalização\b|\blugar\b|\bmancha\b|\bnúcleo\b|\bperiferia\b|\bplaneta\b|\bponto\b|\bre
gião\b|\bregional\b|\bsuperfície\b|\bTerra\b|\bterreno\b|\bterritório\b|\burbano\b|\bzona\b", str(x))))
  df_frame["Localização_da_luz"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacender\b|\bbrilhante\b|\bbrilhar\b|\bbrilho\spálido\b|\bbrilho\b|\bcentelha\b|\bchamejar
\b|\bcintilação\b|\bcintilante\b|\bcintilar\b|\bclaro\b|\bcoruscação\b|\bcoruscar\b|\besplendor\b|\bflameja
r\b|\bflash\b|\biluminado\b|\biluminar\b|\bluminosidade\b|\bluminoso\b|\blustroso\b|\bluz\b|\bpiscante\b|
\bpiscar\b|\brefulgência\b|\brefulgente\b|\brefulgir\b|\breluzir\b|\bresplandecente\b|\bresplandecer\b|\br
esplendor\b|\bsolar\b|\bvislumbrar\b|\bvislumbre\b", str(x))))
  df_frame["Localização_esperada_da_pessoa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcasa\b", str(x))))
  df_frame["Localização_na_trajetória"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bpassar\b",
str(x))))
  df_frame["Localização_no_tempo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bano\b|\bdia\b|\bem\b|\bhora\b|\btempo\b", str(x))))
  df_frame["Localizar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bachar\b|\bencontrar\b",
str(x))))
  df_frame["Louvabilidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bhonra\b", str(x))))
  df_frame["Malfeitoria"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bpecar\b", str(x))))
  df_frame["Maneira"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baltamente\b|\bapaixonadamente\b|\batravés\sde\b|\batravés\b|\bauditivamente\b|\bcin
estesicamente\b|\bcomo\b|\bconforme\b|\bcuriosamente\b|\bde\sum\sjeito\b|\bde\sverdade\b|\bdireta
mente\b|\bdireto\b|\bincontrolavelmente\b|\bintencionalmente\b|\bjeito\b|\blevemente\b|\bliteralmente\b
|\bmaneira\b|\bmaravilhosamente\b|\bmedida\b|\bnormalmente\b|\bobsessivamente\b|\bpoeticamente\
b|\bprofundamente\b|\bprogressivo\b|\bradicalmente\b|\btranquilamente\b|\bvisualmente\b", str(x))))
```

```
  df_frame["Manipulação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapertar\b|\bsegurar\b|\btocar\b", str(x))))
  df_frame["Marca_corporal"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcicatriz\b", str(x))))
  df_frame["Massa_movimento Mass_motion"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafluir\b|\binundar\b", str(x))))
  df_frame["Massa_quantificada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmedida\b|\bmuito\b|\bnenhum\b|\bnúmero\b|\bpeso\b|\btodo\b|\btudo\b", str(x))))
  df_frame["Matar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmatar\b|\bmorto\b", str(x))))
  df_frame["Medida_por_ação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbocado\b|\bpitada\b|\bpunhado\b", str(x))))
  df_frame["Medida_volume"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcolher\sde\ssopa\b|\bfio\b|\bgota\b", str(x))))
  df_frame["Medir_duração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bano\b|\bdia\b|\bhora\b|\bmês\b|\bmilênio\b|\bminuto\b|\bnanossegundo\b|\bquinzena\b
|\bsegundo\b|\bsemana\b|\btempo\b", str(x))))
  df_frame["Meio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batravés\b|\bcanal\b|\bcomo\b|\bcomo\b|\bem\b|\bforma\b|\bjeito\b|\bmecanismo\b|\bm
eio\b|\bmétodo\b|\bmídia\b|\bmodo\sde\soperação\b|\bpor\b|\bprocedimento\b|\bprocesso\b|\breceita\b
|\btática\b|\btécnica\b", str(x))))
  df_frame["Meios_de_comunicação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcelular\b|\btelefone\b", str(x))))
  df_frame["Meios_de_transporte"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bavião\b|\bbalão\b|\bbalsa\b|\bbarca\b|\bbarco\sde\spasseio\b|\bbarco\b|\bbicicleta\b|\b
bonde\b|\bcarro\b|\bfrescão\b|\bhelicóptero\b|\bmetrô\b|\bmotocicleta\b|\bnavio\sde\scruzeiro\b|\bnavio
\b|\bônibus\sde\spasseio\b|\bônibus\b|\bparador\b|\btáxi\saéreo\b|\btáxi\b|\bteleférico\b|\btrailer\b|\btre
m\b|\bvagão\sleito\b|\bveículo\b|\bveleiro\b", str(x))))
  df_frame["Membro_das_forças_armadas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bartilheiro\b|\bcapitão\b|\bnavegador\b", str(x))))
  df_frame["Memória"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\blembrar\b|\brecordação\b",
str(x))))
  df_frame["Mirar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdirigido\b|\bmira\b", str(x))))
  df_frame["Modalidades_esportivas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barremesso\b|\bborboleta\b|\bcanoagem\sslalom\b|\bcanoagem\svelocidade\b|\bciclis
mo\sBMX\scorrida\b|\bciclismo\sBMX\sfreestyle\b|\bciclismo\sBMX\smanobras\b|\bciclismo\sBMX\sra
cing\b|\bciclismo\sBMX\b|\bciclismo\sde\sestrada\b|\bciclismo\sde\spista\b|\bciclismo\smountain\sbike\
b|\bconcurso\scompleto\sde\sequitação\b|\bcorrida\scom\sobstáculos\b|\bcorrida\sde\sfundo\b|\bcorrid
a\sde\slonga\sdistância\b|\bcorrida\sde\svelocidade\b|\bcorrida\b|\bcostas\b|\bcrawl\b|\bcrol\b|\bdecatl
o\b|\bespada\b|\bestilo\slivre\b|\bflorete\b|\bginástica\sartística\b|\bginástica\sde\strampolim\b|\bginásti
ca\srítmica\b|\bheptatlo\b|\bhipismo\sadestramento\b|\bhipismo\sCCE\b|\bhipismo\ssaltos\b|\blançame
nto\b|\bluta\sestilo\slivre\b|\bluta\sgreco-
romana\b|\bmaratona\b|\bmarcha\satlética\b|\bmedley\b|\bmeio-
fundo\b|\bmodalidade\b|\bnado\sborboleta\b|\bnado\scostas\b|\bnado\slivre\b|\bnado\speito\b|\bpark\b|
\bpeito\b|\brevezamento\b|\bsabre\b|\bsalto\scom\svara\b|\bsalto\sem\saltura\b|\bsalto\sem\sdistância\
b|\bsalto\striplo\b|\bsalto\b|\bstreet\b|\btrampolim\sacrobático\b", str(x))))
  df_frame["Modo_de_viver"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baventureiro\b|\bboemia\b|\bboêmio\b|\bdeficiente\b|\bdeficiente\b|\bhippie\b|\bnatureb
a\b|\bnaturismo\b|\bvegano\b|\bvida\b|\bviver\b", str(x))))
  df_frame["Morrer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafogar\b|\baguentar\b|\bfalecer\b|\bresistir\b", str(x))))
  df_frame["Morte"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmorrer\b|\bmorte\b|\bperda\b", str(x))))
  df_frame["Morto_ou_vivo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmortal\b|\bmortal\b|\bvida\b", str(x))))
  df_frame["Móveis"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbanco\b|\bcadeira\b|\bcama\b|\bcarteira\b|\bcolchão\b|\bguarda-
roupa\b|\bmesa\b|\bmóvel\b|\bpoltrona\b|\bprateleira\b|\bsofá-cama\b", str(x))))
  df_frame["Movimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\b|\balterar\b|\bavançar\b|\bbalançar\b|\bbater\b|\bderrubar\b|\bdeslizar\b|\bdesviar\b
|\bdirigir\b|\bdisparar\b|\bempurrar\b|\benrolar\b|\bespiralar\b|\bir\b|\bmover\b|\bmovimento\b|\bmudar\
```

```
b|\bondular\b|\bpercorrer\b|\bpuxar\b|\bremover\b|\brodopiar\b|\brolar\b|\bsair\b|\bseguir\b|\bserpear\b|
\bserpentear\b|\btrançar\b|\bviajar\b|\bvoar\b|\bvolta\b|\bvoltar\b|\bziguezaguear\b", str(x))))
  df_frame["Movimento_corporal"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baberto\b|\bcontorcer\b|\bestender\b|\bfechar\b|\bmexer\b|\bmorder\b|\bmover\b|\bsent
ar\b|\bvirar\b", str(x))))
  df_frame["Movimento_direcional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcair\b|\bpor\b|\bsubmergir\b|\btombo\b", str(x))))
  df_frame["Movimento_fluídico"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcorrente\b|\bfluido\b|\bgota\b", str(x))))
  df_frame["Mudança_de_estado_operacional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapagar\b|\bligar\b|\bligar\b|\bligar\b", str(x))))
  df_frame["Mudança_de_fase"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bgelar\b", str(x))))
  df_frame["Mudança_de_temperatura_incoativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcalor\b", str(x))))
  df_frame["Mudar_direção"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvirar\b", str(x))))
  df_frame["Mudar_duração_do_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bextensão\b|\bperpetuar\b|\bprolongar\b", str(x))))
  df_frame["Mudar_posição_em_uma_escala"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\batingir\b|\bchegar\b|\belevar\b|\bexplosão\b|\btriplicar\b", str(x))))
  df_frame["Mudar_postura"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdeitar\b", str(x))))
  df_frame["Mudar_tempo_do_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdemora\b", str(x))))
  df_frame["Nascer"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bnascer\b|\bnascimento\b",
str(x))))
  df_frame["Negação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bnão\b|\bnunca\b|\bsem\b",
str(x))))
  df_frame["Negar_ou_conceder_permissão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baprovado\b|\baprovar\b", str(x))))
  df_frame["Negócios"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacademia\sde\sdança\sdo\sventre\b|\bacademia\sde\sdança\b|\bacademia\sde\sginá
stica\b|\bacademia\b|\badega\b|\bagência\sde\sentretenimento\b|\bagência\sde\sturismo\b|\bagência\
sde\sviagens\sa\spontos\sturísticos\b|\bagência\b|\bagropecuária\b|\bantiquário\b|\bassistência\smédi
ca\b|\batacadista\b|\bbanco\b|\bbazar\b|\bboate\b|\bboite\b|\bbomboniere\b|\bbookstore\b|\bboutique\b
|\bbutique\b|\bcaixa\seletrônico\b|\bcasa\sde\sdança\b|\bcasa\sde\sshow\b|\bcasa\snoturna\b|\bclub\b|
\bcomércio\sde\spneu\b|\bconcessionária\b|\bconfecção\b|\bconfeitaria\b|\bconsultoria\sde\srecursos\
shumanos\b|\bconsultório\b|\bcorporação\b|\bdelivery\b|\bdesenvolvedora\sde\simóveis\b|\bdestilaria\
b|\bdistribuidor\sde\sbebidas\b|\bdoceria\b|\bdrogaria\b|\beditora\sde\sjornais\b|\beditora\b|\bempreen
dimento\b|\bempresa\sde\slembrancinhas\sde\sfesta\b|\bempresa\sde\sorganização\sde\seventos\b|\
bempresa\sde\svigilância\b|\bempresa\b|\bentrega\sde\srefeições\sprontas\b|\bestabelecimento\b|\bfá
brica\b|\bfarmácia\b|\bfeira\sde\sartesanato\b|\bfirma\b|\bfloricultura\b|\bfornecedor\sde\sartigos\shosp
italares\b|\bfranquia\b|\bfrutaria\b|\bhamburgueria\b|\binvestimento\b|\bjoalheria\b|\bjornal\b|\bkaraokê\
b|\blava-
rápido\b|\blivraria\b|\blocal\scom\smúsica\sao\svivo\b|\blocal\spara\seventos\b|\bloja\sde\sacessórios\
sautomotivos\b|\bloja\sde\sacessórios\sde\smoda\b|\bloja\sde\sartigos\spara\scama\smesa\se\sbanho
\b|\bloja\sde\sartigos\spara\sdança\b|\bloja\sde\sartigos\spara\sfestas\b|\bloja\sde\sazulejos\b|\bloja\s
de\sbrinquedos\b|\bloja\sde\scalçado\b|\bloja\sde\sCDs\susados\b|\bloja\sde\scolchões\b|\bloja\sde\sc
onveniência\b|\bloja\sde\scostura\b|\bloja\sde\sdecoração\b|\bloja\sde\sdepartamento\b|\bloja\sde\sdis
cos\b|\bloja\sde\seletrodomésticos\b|\bloja\sde\seletrônicos\b|\bloja\sde\sjogos\b|\bloja\sde\slingerie\b|
\bloja\sde\smadeiras\b|\bloja\sde\smateriais\sartísticos\b|\bloja\sde\smateriais\sde\sconstrução\b|\bloj
a\sde\smateriais\spara\sartesanato\b|\bloja\sde\smoda\sfeminina\b|\bloja\sde\smoda\sinfantil\b|\bloja\
sde\smoda\smasculina\b|\bloja\sde\smóveis\sinfantis\b|\bloja\sde\smúsica\b|\bloja\sde\spresentes\b|\b
loja\sde\sprodutos\snaturais\b|\bloja\sde\sração\b|\bloja\sde\sroupa\b|\bloja\sde\sroupas\sde\sbanho\b
|\bloja\sde\sroupas\sde\scama\b|\bloja\sde\sroupas\sde\spraia\b|\bloja\sde\sroupas\spara\sbebês\b|\bl
oja\sde\svideogame\b|\bloja\spara\sbebê\b|\bloja\b|\bmercadinho\b|\bmercado\b|\bmercearia\b|\bmulti
nacional\b|\bnegociação\b|\bnegócio\b|\boficina\sde\scarroceria\b|\boperadora\b|\bótica\b|\bperfumari
a\b|\bpet\sshop\b|\bpetshop\b|\bposto\sde\scombustível\b|\bposto\sde\sgasolina\b|\bprodutora\sde\sci
ne\se\svídeo\b|\bpromoção\b|\bprovedor\sde\sinternet\b|\bsalão\sde\sbeleza\b|\bserviço\sde\sajuste\s
de\sroupas\b|\bserviço\sde\salinhamento\se\sbalanceamento\b|\bserviço\spúblico\b|\bserviço\sveterin
ário\sde\semergência\b|\bspa\b|\bsupermercado\b|\bvenda\b|\bvinícola\b|\bwi-fi\b", str(x))))
```

```
  df_frame["Nível_de_luz"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bescuridão\b|\bescuro\b|\bluminoso\b", str(x))))
  df_frame["Nomeação_simples"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bchamar\b",
str(x))))
  df_frame["Nomear"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bnomear\b", str(x))))
  df_frame["Nome_simples"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\boração\b|\bpalavra\b|\bsigla\b|\btermo\b|\bverbo\b|\bvocábulo\b", str(x))))
  df_frame["Notabilidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdestacar-
se\b|\bganhar\b|\bgrande\b|\bmaior\b|\bpequeno\b", str(x))))
  df_frame["Números_cardinais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\b16\b|\b21\b|\bambos\b|\bbilhão\b|\bcatorze\b|\bcem\b|\bcento\b|\bcinco\b|\bcinquenta\
se\sdois\b|\bcinquenta\b|\bdez\b|\bdezenove\b|\bdezesseis\b|\bdezessete\b|\bdezoito\b|\bdois\b|\bdoz
e\b|\bdual\b|\bdupla\b|\bduzentos\b|\bmeio\b|\bmil\b|\bmilhão\b|\bmilhar\b|\bnove\b|\bnoventa\b|\bnúm
ero\b|\boitenta\b|\boito\b|\bonze\b|\bpar\b|\bquarenta\b|\bquatorze\b|\bquatro\b|\bquinhentos\b|\bquinz
e\b|\bseis\b|\bsessenta\b|\bsete\b|\bsetenta\se\squatro\b|\bsetenta\b|\btrês\b|\btreze\b|\btrinta\se\scinc
o\b|\btrinta\se\sdois\b|\btrinta\se\snove\b|\btrinta\se\soito\b|\btrinta\se\squatro\b|\btrinta\se\sseis\b|\btri
nta\se\ssete\b|\btrinta\se\strês\b|\btrinta\se\sum\b|\btrinta\b|\bum\b|\buma\b|\bvinte\se\scinco\b|\bvinte\
se\sdois\b|\bvinte\se\snove\b|\bvinte\se\soito\b|\bvinte\se\squatro\b|\bvinte\se\sseis\b|\bvinte\se\ssete\
b|\bvinte\se\strês\b|\bvinte\se\sum\b|\bvinte\b|\bzero\b|\bzilhão\b", str(x))))
  df_frame["Números_ordinais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdécimo\snono\b|\bdécimo\ssétimo\b|\bdécimo\ssexto\b|\bdécimo\sterceiro\b|\bdécimo
\b|\bdécimo\b|\bnono\b|\bnono\b|\boitavo\b|\boitavo\b|\bprimeiro\b|\bprimeiro\b|\bquarto\b|\bquarto\b|\b
quinto\b|\bsegundo\b|\bsegundo\b|\bsétimo\b|\bsexto\b|\bsexto\b|\bterceiro\b|\bterceiro\b|\búltimo\b",
str(x))))
  df_frame["Obter_documento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdocumento\b|\bobter\b|\brenovar\b|\btirar\b", str(x))))
  df_frame["Obviedade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bclaro\b|\bclaro\b|\bdisponível\b|\bevidente\b|\bimperceptível\b", str(x))))
  df_frame["Ocorrência_condicional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcondicionado\b|\bse\b", str(x))))
  df_frame["Oferecer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\boferecer\b|\boferta\b|\bservir\b", str(x))))
  df_frame["Operar_um_sistema"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfuncionamento\b|\bfuncionar\b|\boperar\b", str(x))))
  df_frame["Operar_veículo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bandar\b|\bmontar\b|\bpilotar\b|\bteleguiado\b", str(x))))
  df_frame["Opinião"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bachar\b|\bacreditar\b|\bcrer\b|\bopinião\b|\bpensar\b|\bteoria\b|\bteoricamente\b|\bvisã
o\b", str(x))))
  df_frame["Oportunidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchance\b|\boportunidade\b|\boportuno\b", str(x))))
  df_frame["Organização"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagência\sde\snotícias\b|\bassociação\b|\bcartel\b|\bclube\b|\bcomitê\b|\bconselho\b|\b
corporação\b|\bdelegação\b|\bdesorganização\b|\bdesorganizar\b|\bempresa\b|\bfraternidade\b|\bgov
erno\b|\bgrupo\b|\binteligência\b|\bjudiciário\b|\bjuntar\b|\bliga\b|\bmultinacional\b|\bordem\b|\borganiz
ação\b|\borganizar\b|\bórgão\b|\bparlamento\b|\bsociedade\b|\bunião\b", str(x))))
  df_frame["Órgão_judicial"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvara\b", str(x))))
  df_frame["Origem"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafricano\b|\bamericano\b|\bárabe\b|\bargentino\b|\basiático\b|\bassírio\b|\bbizantino\b|
\bbrasileiro\b|\bbritânico\b|\bcanadense\b|\bcapixaba\b|\bchinês\b|\bcolombiano\b|\bcubano\b|\bdatar\
b|\bde\b|\begípcio\b|\bescocês\b|\bespanhol\b|\beuropeu\b|\bfinlandês\b|\bfrancês\b|\bgrego\b|\bholân
des\b|\bindiano\b|\bindígena\b|\binternacional\b|\biraniano\b|\biraquiano\b|\birlandês\b|\bitaliano\b|\bja
maicano\b|\bjaponês\b|\bjordaniano\b|\blocal\b|\bmineiro\b|\bnacional\b|\bnacional\b|\boriental\b|\borig
em\b|\botomano\b|\bportuguês\b|\bqueniano\b|\bromano\b|\brusso\b|\bsaudita\b|\bsírio\b|\bsuíço\b|\bt
upinambá\b|\bturco\b|\bvietnamita\b|\bvir\sde\b", str(x))))
  df_frame["Origem_indígena"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bnativo\b", str(x))))
  df_frame["Padrão_temporal"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\britmo\b", str(x))))
  df_frame["Parcialidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bneutro\b", str(x))))
```

```
  df_frame["Parentesco"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagregado\b|\bavó\b|\bfilha\b|\bfilho\b|\birmã\b|\birmão\b|\bmadrinha\b|\bmãe\b|\bmam
ãe\b|\bmaterno\b|\bneto\b|\bpadrasto\b|\bpai\b|\bpapai\b|\bparente\b|\bprimo\b|\btio\b|\bvó\b", str(x))))
  df_frame["Partes_de_roupas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbolso\b|\bbotão\b|\bcapuz\b|\bfita\b|\bsola\b", str(x))))
  df_frame["Partes_do_corpo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbarriga\b|\bboca\b|\bbraço\b|\bcabeça\b|\bcabelo\b|\bcélula\b|\bcérebro\b|\bcintura\b|\
bcolo\b|\bcoluna\b|\bcoração\b|\bcorpo\b|\bcostas\b|\bcostela\b|\bcotovelo\b|\bdedo\b|\bmão\b|\bment
e\b|\bnariz\b|\bolho\b|\bombro\b|\bosso\b|\bpeito\b|\bperna\b|\brosto\b|\btesta\b", str(x))))
  df_frame["Parte_como_segmentos_ordenados"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcamada\b", str(x))))
  df_frame["Parte_elemento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baglomerado\b|\bbocado\b|\bcaco\b|\bchip\b|\bfarrapo\b|\bfatia\b|\bfragmento\b|\bgalho
\b|\blâmina\b|\bmigalha\b|\bnaco\b|\bnódulo\b|\bpedaço\b|\bplaca\b|\btorrão\b|\btrecho\b", str(x))))
  df_frame["Parte_interior_exterior"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexterior\b|\bexterno\b|\binterior\b|\binterno\b", str(x))))
  df_frame["Parte_moldada"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balça\b|\bboca\b|\bborda\b|\bbraço\b|\bcasca\b|\bgraveto\b|\bperna\b", str(x))))
  df_frame["Parte_orientacional"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bápice\b|\bbaixo-
ventre\b|\bbase\b|\bcanto\b|\bcimeira\b|\bcrista\b|\bdireita\b|\bdireito\b|\besquerda\b|\besquerdo\b|\bfa
ce\b|\bfrente\b|\bfrente\b|\bfrontal\b|\bfundo\b|\binferior\b|\binferior\b|\blado\b|\bleste\b|\bleste\b|\bnoro
este\b|\bnorte-
sul\b|\bnorte\b|\bnorte\b|\bocidental\b|\boeste\b|\boeste\b|\boriental\b|\bpé\b|\bpico\b|\bposterior\b|\bret
aguarda\b|\bsubdimensionado\b|\bsul\b|\bsul\b|\bsulista\b|\bsuperior\b|\btopo\b|\btraseiro\b|\bverso\b",
str(x))))
  df_frame["Parte_todo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcentésimo\b|\bcentro\b|\bcompleta\b|\bcompor\b|\bconstituir\b|\bdecomposição\b|\bde
do\b|\bdividir\b|\bfazer\sparte\b|\bfiapo\b|\bformar\b|\bfragmento\b|\bgancho\b|\bgota\b|\bintegrar\b|\bin
teiro\b|\binterno\b|\bmetade\b|\boitavo\b|\bparte\b|\bpertencer\b|\bpingo\b|\bponta\b|\bporção\b|\bprefix
o\b|\bquinto\b|\bseção\b|\bsegmento\b|\bter\b|\bterceiro\b|\btrimestre\b", str(x))))
  df_frame["Participação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbaladar\b|\bcelebrar\b|\bcomprometimento\b|\bintegrante\b|\bjogar\b|\bparticipação\b|\
bparticipante\b|\bparticipar\b", str(x))))
  df_frame["Partida_do_turista_alojamento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcheck-out\b|\bsaída\b", str(x))))
  df_frame["Partida_do_turista_localidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcheck-in\b|\bembarcar\b|\bembarque\b", str(x))))
  df_frame["Partir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafastar\b|\bdebandar\b|\bdeixar\b|\bdesaparecer\b|\bdesaparecimento\b|\bemergir\b|\
bescapar\b|\bexôdo\b|\bfuga\b|\bir\sembora\b|\bir\b|\bpartir\b|\bsaída\b|\bsair\b|\bsumir\b", str(x))))
  df_frame["Partitivo"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bde\b|\bfora\b|\bparte\b",
str(x))))
  df_frame["Peça_arquitetônica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barco\b|\bazulejo\b|\bbalcão\b|\bchão\b|\bcornija\b|\bfachada\b|\bfundação\b|\bjanela\b
|\blaje\b|\blance\b|\blareira\b|\bmurada\b|\bmureta\b|\bparapeito\b|\bparede\b|\bpatamar\b|\bpiso\b|\bp
orta\b|\btelhado\b|\bteto\b|\btrave\b", str(x))))
  df_frame["Pedir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchamar\b|\bconvidar\b|\bconvite\b|\bdemanda\b|\bdemandar\b|\bmandar\b|\bordem\b|
\bpedido\b|\bpedir\b|\bsolicitar\b", str(x))))
  df_frame["Pegar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapanhar\b|\bapossar\b|\bapreender\b|\bapreensão\b|\bcomandar\b|\blevar\b|\bpegar\
b", str(x))))
  df_frame["Pegar_fogo"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bqueimar\b", str(x))))
  df_frame["Percepção_ativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badmirar\b|\barregalar\b|\bassistir\b|\bavistar\b|\bcheirar\b|\bcontemplar\b|\bdeparar\b|\
bembasbacar\b|\bencarar\b|\benxergar\b|\bescutar\b|\bespiar\b|\bespionar\b|\bespreitada\b|\bespreitar
\b|\bfungada\b|\bfungar\b|\bgosto\b|\bobservação\b|\bobservar\b|\bolhar\b|\bolhar\b|\bouvir\b|\bpalpar\
b|\bprovar\b|\brelançar\b|\brelance\b|\bsaborear\b|\bsentir\b|\bver\b", str(x))))
  df_frame["Período_de_tempo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\banoitecer\b|\bdia\sa\sdia\b|\bdia\b|\bhorário\b|\btempo\b|\bvida\b", str(x))))
```

```
  df_frame["Persuasão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconvencer\b|\bmotivar\b", str(x))))
  df_frame["Pessoas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balguém\b|\balguém\b|\bcara\b|\bcaráter\b|\bcavalheiro\b|\bcidadão\b|\bcolega\b|\bcom
panheiro\b|\bdama\b|\bgalera\b|\bgaroto\b|\bgente\b|\bhomem\b|\bhumanidade\b|\bhumano\b|\bindivíd
uo\b|\bmenino\b|\bmoço\b|\bmortal\b|\bmulher\b|\bnenhum\b|\bninguém\b|\bninguém\b|\bpersonagem\
b|\bpessoa\b|\bpessoal\b|\bpovo\b|\bpúblico\b|\bquem\b|\bquem\b|\brapaz\b|\bser\shumano\b|\bser\svi
vo\b|\btodo\smundo\b|\btodos\b|\bum\b|\bvida\b", str(x))))
  df_frame["Pessoas_por_atividade_de_lazer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baventureiro\b|\bbackpacker\b|\bbanhista\b|\bfolião\b|\bfrequentador\b|\bgamer\b|\bgee
k\b|\bjogador\b|\bmotoqueiro\b|\bnaturista\b|\bturista\b|\bviajante\b|\bvisitante\b", str(x))))
  df_frame["Pessoas_por_atividade_transitória"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bobservador\b", str(x))))
  df_frame["Pessoas_por_enquadramento_social"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcaipira\b|\bescravidão\b|\bescravo\b|\bmendigo\b|\bpedinte\b|\bsenhor\b", str(x))))
  df_frame["Pessoas_por_etnia"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bafro-
brasileiro\b|\bbranco\b|\bcigano\b|\bíndio\b|\bnegro\b", str(x))))
  df_frame["Pessoas_por_origem"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balagoano\b|\balemão\b|\bamericano\b|\baustríaco\b|\bboliviano\b|\bbrasileiro\b|\bbrasi
leiro\b|\bbritânico\b|\bcaliforniano\b|\bcarioca\b|\bescocês\b|\bespanhol\b|\bestrangeiro\b|\bET\b|\bfran
cês\b|\bfrancesa\b|\bgrego\b|\bgringo\b|\bholandês\b|\binca\b|\bíndio\b|\binglês\b|\binglesa\b|\biranian
o\b|\birlandês\b|\bitaliano\b|\bmexicano\b|\bnova\siorquino\b|\botomano\b|\bpersa\b|\bportuguês\b|\btu
rco\b", str(x))))
  df_frame["Pessoas_por_religião"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbatista\b|\bbudismo\b|\bbudista\b|\bcandomblé\b|\bcatolicismo\b|\bcatólico\b|\bcristão\
b|\bespírita\b|\bespiritismo\b|\bfanático\b|\bfiel\b|\binfiel\b|\bislamismo\b|\bjudaísmo\b|\bjudeu\b|\blaico\
b|\bmórmon\b|\bmulçumano\b|\bpagão\b|\bprotestante\b|\bprotestantismo\b|\bumbanda\b", str(x))))
  df_frame["Pessoas_por_residência"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvizinho\b",
str(x))))
  df_frame["Pessoas_por_vocação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babade\b|\bacadêmico\b|\badvogado\b|\bagente\sduplo\b|\bagente\b|\balfaiate\b|\balun
o\b|\bambientalista\b|\bantropólogo\b|\bapóstolo\b|\barqueólogo\b|\barquiteto\b|\bartesão\b|\bartista\b|\
bassistente\b|\bastrólogo\b|\bastronauta\b|\batendente\b|\batleta\b|\bator\b|\batriz\b|\bautor\b|\bbancár
io\b|\bbarman\b|\bbeato\b|\bbibliotecário\b|\bbiólogo\b|\bbispo\b|\bbombeiro\b|\bcabeleireiro\b|\bcaçad
or\b|\bcamareiro\b|\bcantor\b|\bcapitão\b|\bcardeal\b|\bcarpinteiro\b|\bcartógrafo\b|\bchefe\sde\scozinh
a\b|\bchefe\b|\bcientista\b|\bcirurgião\splástico\b|\bcomerciante\b|\bcomissário\sde\sbordo\b|\bconcurs
ado\b|\bconsultor\b|\bcontador\b|\bcoreógrafo\b|\bcorrespondente\b|\bcostureiro\b|\bcoveiro\b|\bcozinh
eiro\b|\bcriado\b|\bdançarino\b|\bdedetizador\b|\bdelegado\b|\bdentista\b|\bdeputado\b|\bdesenhista\b|
\bdesenvolvedor\sde\ssoftware\b|\bdesenvolvedor\sweb\b|\bdesigner\b|\bdetetive\sparticular\b|\bdeteti
ve\b|\bdiácono\b|\bdiretor\b|\bdocente\b|\bdono\sde\scasa\b|\beditor\b|\beducador\b|\bempregado\sdo
méstico\b|\bempresário\b|\benfermeiro\b|\bengenheiro\b|\bescritor\b|\bescriturário\b|\bespecialista\b|\b
especulador\b|\bespião\b|\besteticista\b|\bestudante\b|\bexecutivo\b|\bexplorador\b|\bextrativista\b|\bfa
bricante\b|\bfarmacêutico\b|\bfaxineiro\b|\bfazendeiro\b|\bfísico\b|\bfisioterapeuta\b|\bfotógrafo\b|\bfreir
a\b|\bfrentista\b|\bfuncionário\b|\bgaitista\b|\bgandula\b|\bgarçom\b|\bgarçonete\b|\bgarimpeiro\b|\bger
ente\b|\bgovernador\b|\bguarda-
costas\b|\bguia\sturístico\b|\bguia\b|\bhistoriador\b|\binstrumentista\b|\bjardineiro\b|\bjoalheiro\b|\bjorn
alista\b|\bjuiz\b|\blançador\b|\blinguista\b|\bmágico\b|\bmagistrado\b|\bmagnata\sdo\spetróleo\b|\bmala
barista\b|\bmanobrista\b|\bmaqueiro\b|\bmaquinista\b|\bmatemático\b|\bmecânico\b|\bmédico\b|\bmedi
um\b|\bmergulhador\b|\bmineiro\b|\bministro\b|\bmissionário\b|\bmonge\b|\bmonsenhor\b|\bmotoboy\b|
\bmotorista\b|\bmúsico\b|\bneurocientista\b|\boficial\b|\boperador\sde\sturismo\b|\boperário\b|\bpadeir
o\b|\bpadre\b|\bpalestrante\b|\bpalhaço\b|\bparaquedista\b|\bpastor\b|\bpedreiro\b|\bpesquisador\b|\bp
iloto\b|\bpintor\b|\bpirata\b|\bpoeta\b|\bpolícia\scivil\b|\bpolícia\b|\bpolicial\sà\spaisana\b|\bpolicial\b|\bp
olítico\b|\bporta-
voz\b|\bprefeito\b|\bpresbítero\b|\bprodutor\b|\bprofessor\sde\sdança\sde\ssalão\b|\bprofessor\b|\bprof
eta\b|\bprofissional\b|\bprofissional\b|\bprogramador\b|\bpsicólogo\b|\bpsiquiatra\b|\bquímico\b|\bradial
ista\b|\brecepcionista\b|\brecreador\b|\brecreador\b|\brei\smago\b|\brepórter\b|\bsacerdote\b|\bsecretá
rio\b|\bsegurança\b|\bsenador\b|\bseringueiro\b|\bservente\b|\bservidor\b|\bsociologista\b|\bsocorrista\
b|\bsoldado\b|\bsolista\b|\btabelião\b|\btaxista\b|\btécnico\b|\btoxicologista\b|\btrabalhador\b|\buniversit
ário\b|\bvendedor\b|\bveterinário\b|\bvoluntário\b|\bzelador\b", str(x))))
```

```
  df_frame["Planejamento_do_turista"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bplanejamento\b|\bplanejar\b|\bpreparação\b", str(x))))
  df_frame["Plantar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\barborizar\b", str(x))))
  df_frame["Plantas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bangiosperma\b|\bárvore\b|\bcoqueiro\b|\berva\sdaninha\b|\bflor\b|\bflora\b|\bfolha\b|\bf
ruto\b|\bgavinha\b|\bpalmeira\b|\bpau-
brasil\b|\bpoligonáceo\b|\brosa\b|\btrepadeira\b|\btronco\b|\bvara\b", str(x))))
  df_frame["Plenitude"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bvácuo\b", str(x))))
  df_frame["Poder_aquisitivo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmultimilionário\b|\bpobre\b|\brico\b|\briqueza\b", str(x))))
  df_frame["Polícia"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdelegacia\b|\bpolícia\b|\bpoliciamento\b", str(x))))
  df_frame["Popularidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\blegal\b|\bmaneiro\b|\bpopular\b|\bpopularizar\b|\bquente\b", str(x))))
  df_frame["Posição_distribuída"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badornar\b|\balinhar\b|\bcercar\b|\bcobrir\b|\bdecoração\b|\bdecorar\b|\bdesarrumar\b|\
bencapotar\b|\bencher\b|\benfeitar\b|\benvolver\b|\bincrustar\b|\blotar\b|\bornamentar\b|\bpavimentar\b
|\bpontilhar\b|\brecobrir\b|\brevestir\b|\bsobre\b", str(x))))
  df_frame["Posição_em_uma_escala"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\briqueza\b", str(x))))
  df_frame["Posse"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentar\b|\bbem\b|\bcontar\b|\bconter\b|\bcustódia\b|\bde\b|\bdesejo\b|\bdeter\b|\b
dono\b|\bfalta\b|\bfaltar\b|\bfalto\b|\bficar\b|\bfruir\b|\binsuficiente\b|\bmanter\b|\bobter\b|\bpatenteado\
b|\bpatentear\b|\bpertencer\b|\bpertences\b|\bposse\b|\bpossessão\b|\bpossuir\b|\bpropriedade\b|\bpr
oprietário\b|\bpróprio\b|\bquerer\b|\bter\b", str(x))))
  df_frame["Possibilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdar\spara\b|\bdever\b|\bpoder\b|\bprovavelmente\b", str(x))))
  df_frame["Possibilidades"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balternativa\b|\bchance\b|\bdever\b|\bescolha\b|\bfuturo\b|\bmaneira\b|\bopção\b|\bou\
b|\bpoder\b|\bpossível\b|\buso\b", str(x))))
  df_frame["Prática"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bprática\b|\bpraticante\b|\bpraticar\b|\btreinamento\b|\btreinar\b|\btreino\b", str(x))))
  df_frame["Precisão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexato\b|\bicônico\b|\bprecisão\b", str(x))))
  df_frame["Precisar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bnecessidade\b|\bprecisar\b|\bter\sque\b", str(x))))
  df_frame["Prédios"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babrigo\b|\bacrópole\b|\baeroporto\b|\balojamento\b|\bambulatório\b|\banexo\b|\baquár
io\b|\barena\b|\barmazém\b|\barquivo\sestadual\b|\barranha-céu\b|\barranha-
céus\b|\bauditório\b|\bbasílica\b|\bbiblioteca\b|\bcabana\b|\bcâmara\smunicipal\b|\bcampanário\b|\bca
nil\b|\bcaravançará\b|\bcarvoaria\b|\bcasa\sda\sfazenda\b|\bcasa\sde\scampo\b|\bcasa\sde\scultura\b|
\bcasa\sde\sjogos\b|\bcasa\sfluvial\b|\bcasa\b|\bcasebre\b|\bcastelo\b|\bcatedral\b|\bceleiro\b|\bcentro\
scultural\b|\bcentro\sde\sarte\b|\bcentro\sde\sconferências\b|\bcentro\sde\sconvenções\b|\bcentro\sde
\sdiversões\sinfantil\b|\bcentro\sde\seventos\b|\bcentro\sespírita\b|\bcentro\smédico\spúblico\b|\bcentr
o\smédico\b|\bcentro\stecnológico\b|\bchalé\b|\bcidadela\b|\bcine-
theatro\b|\bcinema\b|\bcirco\b|\bclinica\sde\sreabilitação\b|\bclube\sde\sfutebol\b|\bclube\sde\stiro\b|\b
cobertura\b|\bcompanhia\sde\ssaneamento\b|\bcompanhia\steatral\b|\bcondomínio\b|\bconservatório\
b|\bconstrução\b|\bdelegacia\sde\spolícia\b|\bdepartamento\sde\spassaporte\b|\bdepartamento\sde\sp
olícia\sdo\sestado\b|\bdepartamento\sde\ssegurança\spública\b|\bdepartamento\suniversitário\b|\bdep
artamento\b|\bdependência\b|\bdiscoteca\b|\bdomiciliar\b|\bdomicílio\b|\bdormitório\b|\bduplex\b|\bedif
ício\b|\bemergência\b|\bescola\sde\ssamba\b|\bescritório\sde\sempresa\b|\besquadrão\sde\sresgate\b
|\bestábulo\b|\bestação\sde\srádio\b|\bestação\sde\stratamento\sde\ságua\b|\bestação\sferroviária\b|\
bestádio\b|\bestrutura\b|\bestufa\b|\bfábrica\b|\bfarol\b|\bfazenda\b|\bfortaleza\b|\bforte\b|\bfortificação\
b|\bgaleria\sde\sarte\b|\bgaleria\b|\bgalpão\b|\bgaragem\b|\bgazebo\b|\bguarda\smunicipal\b|\bhabitaç
ão\b|\bherdade\b|\bhipódromo\b|\bhospital\sgeral\b|\bhospital\sinfantil\b|\bhospital\smilitar\b|\bhospital\
smunicipal\b|\bhospital\sparticular\b|\bhospital\spsiquiátrico\b|\bhospital\b|\biglu\b|\bigreja\sbatista\b|\bi
greja\b|\bimobiliária\b|\bjardim\sbotânico\b|\bjardim\szoológico\b|\blar\b|\blivraria\b|\bmansão\b|\bmate
rnidade\b|\bmesquita\b|\bmosteiro\b|\bmuseu\sde\sarte\smoderna\b|\bmuseu\sde\sarte\b|\bmuseu\sdo
\spatrimônio\b|\bmuseu\shistórico\slocal\b|\bmuseu\shistórico\b|\bmuseu\smarítimo\b|\bmuseu\smilitar
\b|\bmuseu\b|\bpagode\b|\bpalácio\b|\bparque\sde\sdiversão\b|\bparque\stemático\b|\bpavilhão\sde\se
```

```
ventos\b|\bpavilhão\b|\bpensão\b|\bpetshop\b|\bpinacoteca\b|\bpirâmide\b|\bpoliclínica\b|\bposto\sde\s
saúde\scomunitário\b|\bpraça\b|\bprédio\b|\bprefeitura\b|\bpronto\satendimento\b|\bquartel\b|\bquiosq
ue\b|\brepartição\spública\smunicipal\b|\bresidência\b|\brotunda\b|\bruína\b|\bsala\sde\sconcertos\b|\b
salão\sde\sdança\b|\bsalão\sde\sfesta\b|\bsalão\b|\bsauna\sgay\b|\bsauna\b|\bsecretaria\smunicipal\s
de\ssegurança\b|\bsecretaria\smunicipal\sdo\smeio\sambiente\b|\bserviço\sde\ssaúde\smental\b|\bsh
opping\scenter\b|\bshopping\b|\bsinagoga\b|\bsolar\b|\bsupermercado\b|\bteatro\b|\btemplo\b|\btenda\
síndia\b|\btenda\b|\btermas\b|\bterminal\b|\bteto\b|\btorre\b|\btriplex\b|\bvila\b|\bzoo\b|\bzoológico\b",
str(x))))
  df_frame["Preencher"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babastecer\b|\blotar\b|\bpintar\b", str(x))))
  df_frame["Preferência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpreferência\b|\bpreferir\b|\bpreterir\b", str(x))))
  df_frame["Preferred_alternative_scenario"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfavorito\b|\bpreferido\b", str(x))))
  df_frame["Preliminares"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baclimatação\b|\baclimatar\b|\bconcentração\b|\bconcentrar\b", str(x))))
  df_frame["Prendedor"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\badesivo\b|\blacre\b",
str(x))))
  df_frame["Prender"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdomiciliar\b|\bprender\b|\bprisão\b", str(x))))
  df_frame["Presença"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barraigado\b|\bfaltar\b|\bmanifesto\b|\bpresente\b", str(x))))
  df_frame["Presságio"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bprenunciar\b|\bprenúncio\b", str(x))))
  df_frame["Prevaricação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmentir\b", str(x))))
  df_frame["Primeiro_na_classificação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bprincipalmente\b", str(x))))
  df_frame["Probabilidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bprobabilidade\b",
str(x))))
  df_frame["Processo_continuar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcorrer\b|\bficar\b|\bproceder\b", str(x))))
  df_frame["Processo_estado_completo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcompleto\b", str(x))))
  df_frame["Processo_iniciar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomeçar\b|\berupção\b|\bestrear\b|\binaugurar\b|\bincipiente\b|\biniciar\b|\binício\b|\bir
romper\b|\bnascente\b|\bpassar\b|\bprincipiar\b|\bsurgimento\b", str(x))))
  df_frame["Processo_nuclear"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bradioatividade\b",
str(x))))
  df_frame["Processo_parar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcessar\b", str(x))))
  df_frame["Procurar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbusca\b|\bbuscar\b|\bprocurado\b", str(x))))
  df_frame["Profissionais_médicos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\benfermeiro\b", str(x))))
  df_frame["Progression"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdesenvolver\b|\bprogressivamente\b", str(x))))
  df_frame["Prohibiting_or_licensing"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badmitir\b|\baprovar\b|\bdeixar\b|\bpermitir\b|\bproibir\b", str(x))))
  df_frame["Projeto"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bestratégia\b|\bplanejar\b|\bprograma\b|\bprojeto\b", str(x))))
  df_frame["Propor_ideia"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bprojetar\b", str(x))))
  df_frame["Propriedade_mental"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babstrato\b|\bartista\b|\bbagunceiro\b|\bbom\b|\bbrilhante\b|\bcriatividade\b|\bcriativo\b|
\bcuidado\b|\bcuidadoso\b|\bcurioso\b|\bdoido\b|\bexcepcional\b|\bfilosófico\b|\bgenial\b|\bgenialidade\
b|\bhumorado\b|\binteligência\b|\blouco\b|\bresponsável\b|\bsensível\b|\bsolícito\b|\btalentoso\b|\btími
do\b|\bvergonha\b", str(x))))
  df_frame["Prosperar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcrescer\b", str(x))))
  df_frame["Prova"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bteste\b", str(x))))
  df_frame["Proximidade_graduável"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafastado\b|\blonge\b|\bproximidade\b|\bpróximo\b|\bpróximo\b|\brente\b", str(x))))
```

```
  df_frame["Proximidade_não_graduável"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badiante\b|\bao\slado\sde\b|\batrás\sde\b|\bdebaixo\sde\b|\bem\sfrente\sde\b|\bembaix
o\sde\b|\bperto\sde\b|\brente\sa\b|\bsob\b", str(x))))
  df_frame["Publicar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\blançamento\b|\bpublicar\b",
str(x))))
  df_frame["Quadro_de_horários"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagendamento\b|\bprogramação\b|\broteiro\b", str(x))))
  df_frame["Qualidades_de_cor"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmonocromático\b|\bpálido\b|\bvibrante\b", str(x))))
  df_frame["Quantidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcento\b|\bdiverso\b|\bmais\sou\smenos\b|\bmais\b|\bmenos\b|\bmuito\b", str(x))))
  df_frame["Quantidade_proporcional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baproximadamente\b|\baté\b|\bcerca\sde\b|\bmais\b|\bmuito\b|\bpouco\b|\bpouquinho\
b|\bpraticamente\b|\bquase\b|\bvários\b", str(x))))
  df_frame["Quebrar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrebentar\b|\bdescolar\b|\bquebrar\b|\bsoltar\b", str(x))))
  df_frame["Queimar_com_fogo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfogo\b|\bfogueira\b", str(x))))
  df_frame["Questionar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdúvida\b|\bperguntar\b|\bquestionamento\b|\bquestionar\b", str(x))))
  df_frame["Razão"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpor\sisso\b|\bpor\sque\b|\bpor\b|\bporquê\b|\bprincípio\b|\brazão\b|\bsentido\b",
str(x))))
  df_frame["Reações_da_torcida"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baplaudir\b|\bapoiar\b|\bassistir\b|\bchorar\b|\bcomemoração\b|\bcomemorar\b|\bdespe
rtar\b|\bexplodir\b|\bfrustração\b|\bgritar\b|\bobservar\b|\bovacionar\b|\btorcer\b|\bvaiar\b|\bver\b|\bvibr
ar\b", str(x))))
  df_frame["Realização"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balcançar\b|\bconquista\b|\brealização\b", str(x))))
  df_frame["Recipientes"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bburaco\b|\bcaixa\b|\bcesta\b|\bchopeira\b|\bcompartimento\b|\bcopo\b|\bcumbuca\b|\b
embalagem\b|\bgaiola\b|\bgarrafa\b|\bpá\b|\bpoço\b|\bpote\b|\bsacola\b|\bvaso\b", str(x))))
  df_frame["Reclamar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bqueixa\b|\breclamação\b",
str(x))))
  df_frame["Rede"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brede\b|\bweb\b", str(x))))
  df_frame["Referir-se_pelo_nome"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bchamar\b|\bdesignação\b|\bendereçar\b|\bnome\b|\breferir\b", str(x))))
  df_frame["Registro"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcópia\b|\bedição\b|\bobra\b", str(x))))
  df_frame["Relação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\be\b|\bligação\b|\brelação\b", str(x))))
  df_frame["Relação_de_duração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdurante\b|\bdurar\b|\bperdurar\b|\bpor\b", str(x))))
  df_frame["Relação_de_perfilamento_interior"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\b|\bdentre\b|\bdentro\sde\b|\bdentro\b|\bem\smeio\sa\b|\bem\b|\bentre\b|\bexterno\b
|\bfora\b|\binterno\b|\bno\sinterior\sde\b|\bno\smeio\sde\b", str(x))))
  df_frame["Relação_locativa"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bà\sfrente\sde\b|\bacima\sdo\ssolo\b|\bacima\b|\badjacente\b|\balém\sde\b|\balhures\b|
\bali\b|\bao\slongo\sde\b|\baonde\b|\baqui\b|\baté\b|\batravés\sde\b|\bcá\b|\bcontinental\b|\bcontinenta
l\b|\bdepois\b|\bdistante\b|\bem\stoda\sparte\b|\bem\stodo\b|\bem\b|\bembaixo\b|\bencontrar\b|\bentre\
b|\benvolver\b|\bfora\b|\bfronteirar\b|\blá\b|\blonge\b|\bno\sar\b|\bno\stopo\b|\bonde\b|\bonipresente\b|\
bpara\scima\b|\bpara\b|\bparalelo\sa\b|\bperto\b|\bremoto\b|\bsobre\b|\bsubterrâneo\b", str(x))))
  df_frame["Relação_locativa_direcional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babaixo\b|\bacima\b|\bem\scima\b|\bem\sfrente\b|\bfora\sde\b|\bleste\b|\bleste\b|\bnord
este\b|\bnordeste\b|\bnoroeste\b|\bnoroeste\b|\bnorte\b|\bnorte\b|\boeste\b|\boeste\b|\bsudeste\b|\bsu
deste\b|\bsudoeste\b|\bsudoeste\b|\bsul\b|\bsul\b", str(x))))
  df_frame["Relações_pessoais"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacompanhante\b|\badultério\b|\bafastado\b|\bamado\b|\bamante\b|\bamigar-
se\b|\bamigo\b|\bamizade\b|\bamoroso\b|\barrumar\b|\bcamarada\b|\bcasado\b|\bcasal\b|\bcasamento
\b|\bcaso\b|\bcoabitação\b|\bcoabitar\b|\bcolega\b|\bcompanheirismo\b|\bcompanheiro\b|\bconjugal\b|\
```

```
bcortejar\b|\bdivorciado\b|\bdivorciado\b|\bdormir\scom\b|\benamorada\b|\bencontrar\b|\benviuvar\b|\b
esposa\b|\besposo\b|\besposo\b|\bfamília\b|\bfamiliar\b|\bhomoafetivo\b|\bíntimo\b|\bmarido\b|\bnamor
ado\b|\bnamorado\b|\bnamorador\b|\bnamoro\b|\bnoiva\b|\bnoivado\b|\bnoivo\b|\bnoivo\b|\bpaquera\b|
\bparceiro\b|\bparceria\b|\bpegação\b|\bpretendente\b|\brameira\b|\brelação\b|\brelacionamento\b|\bro
mance\b|\bsolteirão\b|\bsolteiro\b|\bsolteirona\b|\btérmino\b|\btraição\b|\bviúva\b|\bviúvo\b|\bviúvo\b",
str(x))))
  df_frame["Remover"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdedetização\b|\bextrair\b|\blavagem\b|\blavar\b|\bpré-
lavagem\b|\bremoção\b|\bremover\b|\bretirar\b|\btirar\b", str(x))))
  df_frame["Reparação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcompensar\b", str(x))))
  df_frame["Representação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpontuar\b|\bselo\b|\bsimbolismo\b|\bsimbolo\b", str(x))))
  df_frame["Representantes"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brepresentante\b",
str(x))))
  df_frame["Request_entity"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpedido\b|\bpedir\b|\bsolicitação\b|\bsolicitar\b", str(x))))
  df_frame["Resgatar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bresgatar\b|\bsalvar\b",
str(x))))
  df_frame["Residência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacampado\b|\bacampamento\b|\bacampar\b|\bcampista\b|\bcolega\sde\squarto\b|\bfic
ar\b|\bhabitado\b|\bhabitante\b|\bhabitar\b|\bhospedar\b|\blocatário\b|\bmorador\b|\bmorar\b|\bocupant
e\b|\bocupar\b|\bradicar\b|\bresidente\b|\bresidir\b|\bviver\b", str(x))))
  df_frame["Resolver_problema"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconserto\b|\bresolver\b", str(x))))
  df_frame["Respirar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bofegar\b|\brespiração\b|\bsuspirar\b", str(x))))
  df_frame["Responsibility"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bresponsável\b",
str(x))))
  df_frame["Resto"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bresto\b", str(x))))
  df_frame["Restringir_movimento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcativar\b|\breclusão\b", str(x))))
  df_frame["Resumir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bresumir\b", str(x))))
  df_frame["Retaining"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bguardar\b|\brealizar\b",
str(x))))
  df_frame["Retirar_da_participação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bwithdraw\b", str(x))))
  df_frame["Reunir-se"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconcentrar\b|\benglobar\b|\breencontro\b|\breunir\b", str(x))))
  df_frame["Revelar_secredo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconfessar\b|\bdesvendar\b", str(x))))
  df_frame["Roubo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babstração\b|\babstrair\b|\bapropriação\sindevida\b|\barrastão\b|\bassalto\b|\bbatedor\
sde\scarteira\b|\bbater\scarteira\b|\bde\sdedos\sleves\b|\bdesviar\b|\bdesvio\b|\bfurtar\b|\bfurto\b|\blad
rão\b|\bpropina\b|\broubado\b|\broubar\b|\broubo\b", str(x))))
  df_frame["Sair_de_um_lugar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babandonar\b|\bdeixar\b|\bdemitir\b|\bdeserção\b|\bdesertar\b|\bdesocupar\b|\bemigra
ção\b|\bemigrante\b|\bemigrar\b|\bexilado\b|\bfugir\b|\bfugitivo\b|\binvestir\b|\bremover\b|\bretirada\b|\b
retirar\b|\bseparar\b", str(x))))
  df_frame["Sair_do_emprego"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\baposentar\b",
str(x))))
  df_frame["Sanções"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badvertência\b|\bcartão\samarelo\b|\bcartão\spreto\b|\bcartão\svermelho\b|\bcartão\b|\
bdesclassificação\b|\bdesqualificação\b|\bexclusão\b|\bexpulsão\b|\bimpedimento\b|\binelegibilidade\b
|\blance-livre\b|\blateral\b|\bman-
up\b|\bpasse\sà\sfrente\b|\bpassividade\b|\bpena\b|\bpenalidade\b|\bpênalti\b|\bsanção\b|\bsuspensão
\b|\btiro\slivre\b|\bvantagem\b", str(x))))
  df_frame["Satisfazer"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\batender\b|\bsatisfazer\b",
str(x))))
  df_frame["Scheduling"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagendamento\b|\bagendar\b", str(x))))
```

```
  df_frame["Sediar_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\borganizar\b|\bpaís\ssede\b|\breceber\b|\bsede\b|\bsediar\b", str(x))))
  df_frame["Semelhança"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcomum\b", str(x))))
  df_frame["Sensação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bmau-
cheiro\b|\bperfume\b|\bsabor\b|\bsensação\b|\bsentir\b|\bvista\b", str(x))))
  df_frame["Sentir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bastral\b|\bauto-
estima\b|\bcalma\b|\bconsolo\b|\bemoção\b|\bentusiasmo\b|\bexperienciar\b|\bira\b|\borgulho\b|\bpaz\
b|\bprazer\b|\bsensação\b|\bsentimento\b|\bsentir\b|\btranquilizar\b", str(x))))
  df_frame["Sequência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bordem\b|\bseguido\b|\bsequência\b|\bsérie\b|\búltimo\b", str(x))))
  df_frame["Serviço_em_alimentação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacepipe\b|\bacompanhamento\b|\bacompanhar\b|\baperitivo\b|\bcafé\sda\smanhã\b|\
bcafé\b|\bcoffee\sbreak\b|\bcoffee-break\b|\bentrada\b|\bpetisco\b|\bpetit-
déjeuner\b|\bprato\sprincipal\b|\bserviço\b|\bservir\b|\bsobremesa\b|\btira-gosto\b", str(x))))
  df_frame["Serviço_turístico"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balmoço\b|\bcafé\sda\smanhã\b|\bcity\stour\b|\bdispor\b|\bjantar\b|\blanche\b|\bmeia\s
pensão\b|\bpacote\sturístico\b|\bpacote\b|\bpensão\scompleta\b|\bpetit-
déjeuner\b|\brefeição\b|\bserviço\b|\btraslado\b", str(x))))
  df_frame["Serviço_turístico_comprar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomprar\b|\bcontratar\b", str(x))))
  df_frame["Serviço_turístico_pagar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bpagamento\b|\bpagar\b", str(x))))
  df_frame["Serviço_turístico_receber"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcobrar\b",
str(x))))
  df_frame["Serviço_turístico_reservar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\breserva\b|\breservar\b", str(x))))
  df_frame["Serviço_turístico_vender"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagência\b|\bdisponibilizar\b|\boferecer\b|\boperador\b|\bproporcionar\b|\bter\b|\bvende
r\b", str(x))))
  df_frame["Ser_afetado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bafetar\b|\bter\b",
str(x))))
  df_frame["Ser_apto"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bapto\b|\bsuficiente\b",
str(x))))
  df_frame["Ser_empregado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btrabalhar\b",
str(x))))
  df_frame["Ser_localizado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bencontrar\b|\bestar\b|\bficar\b|\blocalizado\b|\blocalizar\b|\bparadeiro\b|\bsituado\b|\bs
ituar\b", str(x))))
  df_frame["Ser_necessário"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bessencial\b|\bexigido\b|\bindispensável\b|\bnecessário\b|\bnecessidade\b|\bnecessitar
\b|\brequerido\b|\brequerimento\b", str(x))))
  df_frame["Ser_nomeado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapelido\b|\bchamado\b|\bconhecido\scomo\b|\bconhecido\b", str(x))))
  df_frame["Ser_obrigado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bafazer\b|\bdever\b|\bobrigar\b|\btarefa\b", str(x))))
  df_frame["Ser_obrigatório"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcumprir\b|\bdever\b|\bexigência\b|\bfundamental\b|\bimprescindível\b|\bindispensável\
b|\bmandatório\b|\bobrigar\b|\bobrigatoriamente\b|\bobrigatório\b|\brequisito\b|\bvital\b", str(x))))
  df_frame["Ser_operacional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\badiantar\b|\bfunção\b|\bfuncional\b|\bfuncionar\b|\boperacional\b|\bquebrado\b",
str(x))))
  df_frame["Ser_relevante"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brelevante\b", str(x))))
  df_frame["Sex"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcruzamento\b|\bcruzar\b",
str(x))))
  df_frame["Sharing"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcompartilhar\b|\bcompartilhável\b|\bdividido\b|\bdividir\b", str(x))))
  df_frame["Simultaneidade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bco-
ocorrência\b|\bco-
ocorrer\b|\bcoincidir\b|\bconcorrência\b|\bconcorrente\b|\bconjunção\b|\bsimultaneamente\b|\bsimultan
eidade\b|\bsimultâneo\b", str(x))))
```

```
  df_frame["Sinal"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bindicar\b", str(x))))
  df_frame["Sinceridade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdissimulado\b|\boblíquo\b|\bsincero\b", str(x))))
  df_frame["Sistema"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\besquema\b|\bestrutura\b|\bsistema\b", str(x))))
  df_frame["Sobreviver"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bviver\b", str(x))))
  df_frame["Sofrer_mudança"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\balternar-
se\b|\bdesabar\b|\bguinar\b|\binexorável\b|\binstável\b|\bir\b|\bmudança\b|\bmudar\b|\boscilar\b|\btran
sição\b|\btransição\b|\btroca\b|\btrocar\b|\bvirar\b|\bvoltar\b", str(x))))
  df_frame["Sofrer_transformação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bconversão\b|\bconverter\b|\bdeixar\b|\btornar\b|\btransformar\b|\btransição\b|\btransm
utação\b|\btransmutar\b|\btransubstanciação\b|\btransubstanciar\b", str(x))))
  df_frame["Sons"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacústica\b|\bbarulhento\b|\bestalo\b|\bsom\b|\bvoz\b", str(x))))
  df_frame["Spatial_contact"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bao\slado\b|\bcontato\b|\bcontra\b|\bem\scima\b|\bem\b|\bfazer\scontato\b|\bíngreme\b
|\bno\stopo\b|\bsobre\b|\btangente\b|\btocar\b", str(x))))
  df_frame["Status_de_sigilo"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bsegredo\b",
str(x))))
  df_frame["Sub-região_temporal"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcomeço\b|\bfim\b|\bfinal\b|\binício\b|\binício\b|\bintermediário\b|\bmarço\b|\bmeio\b|\bp
osterior\b|\bprévio\b|\bprincípio\b|\btardio\b|\bvirada\b", str(x))))
  df_frame["Subordinados_e_superiores"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bhierarquia\b", str(x))))
  df_frame["Subpartes_de_artefato"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcabo\b|\bHD\b|\btela\b", str(x))))
  df_frame["Subpartes_de_instalações_esportivas"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balto-
falante\b|\balvo\b|\barco\sde\strês\spontos\b|\bárea\sde\saterrissagem\b|\bárea\sde\slançamento\b|\b
área\sde\slance\slivre\b|\bárea\sde\squeda\b|\bárea\sde\ssaque\b|\bárea\sinterna\sdo\sgol\b|\bárea\sr
estritiva\b|\bárea\stécnica\b|\bárea\b|\baro\b|\barquibancada\b|\bassento\b|\bbaliza\b|\bbanco\sde\sres
ervas\b|\bbanco\b|\bbandeira\sde\sescanteio\b|\bbandeira\b|\bbanheiro\b|\bbarra\b|\bbarreira\b|\bbase
\sdo\srebatedor\b|\bbilheteria\b|\bbloco\sde\spartida\b|\bbloco\sinicial\b|\bcabine\sde\stransmissão\b|\
bcaixa\sde\sareia\b|\bcaixa\sde\saterrissagem\b|\bcaixa\sdo\stécnico\b|\bcamarote\b|\bcesta\b|\bcírcul
o\scentral\b|\bcírculo\sde\sdisparo\b|\bcírculo\sde\slançamento\b|\bcolchão\b|\bencaixe\b|\bescanteio\
b|\bfaixa\b|\bfosso\scom\ságua\b|\bfosso\sde\ságua\b|\bgaiola\b|\bgarrafão\b|\bgol\b|\bgramado\b|\bgr
ande\sárea\b|\bgrande-
área\b|\blateral\b|\blimite\sde\scampo\sexterno\b|\blimite\sde\scampo\sinterno\b|\blinha\scentral\b|\blin
ha\sda\sgrande\sárea\b|\blinha\sda\spequena\sárea\b|\blinha\sde\sarremesso\slateral\b|\blinha\sde\sa
rremesso\slivre\b|\blinha\sde\sataque\b|\blinha\sde\sdez\smetros\b|\blinha\sde\sfalta\b|\blinha\sde\sfu
ndo\b|\blinha\sde\sgol\b|\blinha\sde\slance\slivre\b|\blinha\sde\smeio-
campo\b|\blinha\sde\srestrição\sdo\sgoleiro\b|\blinha\sde\ssaque\b|\blinha\sde\sseis\smetros\b|\blinha\
sde\ssete\smetros\b|\blinha\sde\stiro\slivre\b|\blinha\sde\strês\spontos\b|\blinha\sde\svinte\se\sdois\s
metros\b|\blinha\sde\svinte\se\strês\smetros\b|\blinha\sde\szona\smorta\b|\blinha\slateral\b|\bmarca\sc
entral\b|\bmarca\sde\spênalti\b|\bmastro\sde\sfalta\b|\bmeia-lua\b|\bmeio\sde\scampo\b|\bmeio-
campo\b|\bmesa\b|\bmonte\sdo\slançador\b|\bobstáculo\b|\bpequena\sárea\b|\bpiscina\b|\bpista\sde\s
alerta\b|\bpista\b|\bplacar\seletrônico\b|\bplacar\b|\bplataforma\b|\bposte\b|\bprimeira\sbase\b|\bquarta
\sbase\b|\braia\b|\brampa\b|\brede\b|\bringue\b|\bsarrafo\b|\bsegunda\sbase\b|\bsetor\b|\bstriking\scirc
le\b|\btabela\b|\btablado\b|\btábua\sde\simpulsão\b|\btábua\sde\ssalto\b|\btapete\b|\btatame\b|\btelão\
b|\bterceira\sbase\b|\btoalete\b|\btoilette\b|\btrave\b|\btravessão\b|\btry\sline\b|\bzona\sde\saterrissage
m\b|\bzona\sde\sdois\spontos\b|\bzona\sde\slançamento\b|\bzona\sde\spassagem\b|\bzona\sde\sserv
iço\b|\bzona\sde\strás\b|\bzona\sfrontal\b|\bzona\slivre\b", str(x))))
  df_frame["Subpartes_de_prédios"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacademia\b|\badega\b|\bala\b|\baltar\b|\bandar\b|\bante-
sala\b|\bantecâmara\b|\bapartamento\b|\bárea\sde\slazer\b|\bárea\sde\sserviço\b|\bárea\b|\batelier\b|\
bbanheiro\sprivativo\b|\bbanheiro\b|\bbar\b|\bberçário\b|\bbrinquedoteca\b|\bcâmara\b|\bcampanário\b
|\bcantina\b|\bcapela\b|\bcarrinho\b|\bcatacumba\b|\bcela\b|\bcerca\b|\bchancelaria\b|\bchão\b|\bclose
t\b|\bcômodo\b|\bcopa\b|\bcorredor\b|\bcozinha\b|\bdepósito\sde\sbagagem\b|\bdepósito\b|\bdespens
a\b|\belevador\spanorâmico\b|\belevador\b|\bescada\b|\bescritório\b|\bestúdio\b|\blavabo\b|\blavanderi
a\b|\blavatório\b|\bludoteca\b|\boficina\b|\bpiscina\b|\bporão\b|\bpresbitério\b|\bquadra\sde\stênis\b|\bq
```

```
uadra\b|\bquarto\sde\shóspedes\b|\bquarto\sprincipal\b|\bquarto\b|\bquitinete\b|\brefeitório\b|\brefúgio\
b|\brestaurante\b|\bsacada\b|\bsacristia\b|\bsaguão\b|\bsala\sde\sestar\b|\bsala\sde\sestudos\b|\bsala\
sde\sjantar\b|\bsala\sde\sTV\b|\bsala\b|\bsalão\b|\bsauna\b|\bsolário\b|\bsótão\b|\bspa\b|\bsubsolo\b|\
bterraço\b|\btérreo\b|\bteto\b|\btoalete\b|\btoilette\b|\btorre\b|\bvaranda\b|\bvestiário\b|\bvestíbulo\b",
str(x))))
  df_frame["Subpartes_de_veículos"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bmotor\b|\bvolante\b", str(x))))
  df_frame["Substâncias"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bágua\b|\bamaciante\b|\bar\b|\bareia\b|\bátomo\b|\bborracha\b|\bbronze\b|\bcascalho\b
|\bdetergente\b|\bdiamante\b|\belétron\b|\bferro\b|\bincenso\b|\blágrima\b|\blátex\b|\bmadeira\b|\bmas
sinha\b|\bmatéria\b|\bmaterial\b|\bmetal\b|\bmineral\b|\bminério\b|\bmirra\b|\bmolécula\b|\bmonazite\b|
\borgânico\b|\bouro\b|\bpapel\b|\bpedra\b|\bpirita\b|\bplástico\b|\bpoeira\b|\bpoluição\b|\bsangue\b|\bs
ubstância\b|\bterra\b", str(x))))
  df_frame["Substância_por_fase"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\blíquido\b|\bsólido\b|\bviscoso\b", str(x))))
  df_frame["Sucesso_ou_falha"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbem\ssucedido\b|\bconseguir\b|\bfracassar\b|\bperder\sgol\b|\bperder\b|\brodar\b",
str(x))))
  df_frame["Suficiência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babundante\b|\badequação\b|\badequadamente\b|\badequado\b|\bamplo\b|\bbastante\
b|\bbastar\b|\bdemais\b|\bfartura\b|\binadequação\b|\binadequadamente\b|\binadequado\b|\binsuficiên
cia\b|\binsuficiente\b|\binsuficientemente\b|\bsem\b|\bser\ssuficiente\b|\bservir\b|\bsuficiente\b|\bsuficie
ntemente\b|\btanto\b", str(x))))
  df_frame["Superar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bpassar\b|\bultrapassar\b",
str(x))))
  df_frame["Tamanho"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\balto\b|\bamplo\b|\bcolossal\b|\bdiminuto\b|\benorme\b|\bespaçoso\b|\bestrondoso\b|\b
gigante\b|\bgigantesco\b|\bgrande\b|\bimenso\b|\bimensurável\b|\bínfimo\b|\binfinitesimal\b|\bjumbo\b|
\bligeiro\b|\bliliputiano\b|\bmaior\b|\bmassivo\b|\bmediano\b|\bmédio\b|\bmeio-
metro\b|\bmenor\b|\bmini\b|\bminiatura\b|\bminúsculo\b|\bpequeno\b|\bsubstancial\b|\bvolumoso\b",
str(x))))
  df_frame["Temer"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bamedrontado\b|\bapreensão\b|\bassustado\b|\baterrorizado\b|\blevar\ssusto\b|\bmedo
\b|\bnervoso\b|\bpavor\b|\bsurtado\b|\bterror\b|\bviver\scom\smedo\b", str(x))))
  df_frame["Temeridade"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bpaciente\b", str(x))))
  df_frame["Temperatura"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcongelante\b|\bescaldante\b|\bfresco\b|\bfrio\b|\bgelado\b|\bmorno\b|\bquente\b|\btem
peratura\b|\btépido\b", str(x))))
  df_frame["Temperatura_ambiente"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\babafado\b|\bcongelante\b|\bfresco\b|\bfrio\b|\bfrio\b|\bmorno\b|\bquente\b|\btemperatu
ra\b", str(x))))
  df_frame["Tempo_período_de_ação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdemorado\b|\bdemorar\b|\bdia\sa\sdia\b|\bdia\sa\sdia\b", str(x))))
  df_frame["Tempo_relativo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\banterior\b|\bantigamente\b|\bantiguidade\b|\batrasado\b|\batualidade\b|\bcedo\b|\bcon
secutivo\b|\bdepois\b|\benquanto\b|\bpassado\b|\bpróximo\b|\brecente\b|\bseguido\b|\btarde\b|\búltimo
\b", str(x))))
  df_frame["Tentar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bdedicação\b|\bentregar\b|\besforçar\b|\btentar\b|\btentativa\b", str(x))))
  df_frame["Tentar_persuadir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baconselhar\b|\bconselho\b|\bpor\scontra\sa\sparede\b|\bpressionar\b|\brecomendaçã
o\b|\brecomendado\b|\brecomendar\b", str(x))))
  df_frame["Terminar_competição"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bperdedor\b|\bvencedor\b|\bvitória\b", str(x))))
  df_frame["Ter_associado"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcom\b", str(x))))
  df_frame["Ter_ou_carecer_de_acesso"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacesso\b", str(x))))
  df_frame["Ter_visita"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bconvidar\b", str(x))))
  df_frame["Teste_de_operação"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btestar\b",
str(x))))
```

```
  df_frame["Texto"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bapresentação\b|\bartigo\b|\bcardápio\b|\bcartão\b|\bcartaz\b|\bcatálogo\b|\bcriação\spl
ástica\b|\bdesenho\b|\bebook\b|\bfrase\b|\bguia\b|\bhistória\b|\binstrução\b|\bjornal\b|\blenda\b|\blingu
agem\b|\blista\b|\bliteratura\b|\blivro\b|\bmapa\b|\bmenu\b|\bnarrativa\b|\bobra\b|\bpoema\b|\bpost\b|\b
postal\b|\bprosa\b|\bprovérbio\b|\bquadrinho\b|\brascunho\b|\breportagem\b|\bsentença\b|\bteatro\b|\bt
exto\b", str(x)))
  df_frame["Texto_criação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bautobiografia\b|\bbiografia\b|\bcriação\stextual\b|\bcrônica\b|\bescrever\b|\blegendar\b
|\bpoesia\b|\breportagem\b|\btexto\b", str(x)))
  df_frame["Tipicalidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcurioso\b|\bespecífico\b|\bestranho\b|\bparticular\b|\bprecioso\b|\btipicamente\b|\btípic
o\b", str(x)))
  df_frame["Tipo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bde\b|\bespécie\b|\bmodo\b|\btipo\b", str(x)))
  df_frame["Tomar_forma"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcurvar\b|\bdobrar\b|\benrolar\b|\btorcer\b", str(x)))
  df_frame["Tomar_partido"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\sfavor\b|\bcontra\b|\bcontra\b|\bluta\b", str(x)))
  df_frame["Tópico"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\srespeito\sde\b|\babordar\b|\bassunto\b|\bponto\b|\bsobre\b|\btema\b|\btópico\b",
str(x)))
  df_frame["Torcida"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bespectador\b|\bfã\b|\bplateia\b|\btelespectador\b|\btorcedor\b|\btorcida\b", str(x)))
  df_frame["Tornar-se"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bficar\b", str(x))))
  df_frame["Tornar-se_consciente"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bachar\b|\bciente\b|\bdescoberta\b|\bdescoberto\b|\bdescobrimento\b|\bdescobrir\b|\bd
esmascarar\b|\bdetectar\b|\bdiscernir\b|\bdizer\b|\bencontrar\b|\bespionar\b|\bnotar\b|\bobservar\b|\bol
har\b|\bperceber\b|\breconhecer\b|\breconhecimento\b|\bregistrar\b", str(x)))
  df_frame["Tornar-se_membro"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bingresso\b",
str(x))))
  df_frame["Tornar-se_não-operacional"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bfurar\b|\bquebrar\b|\bqueimar\b|\brasgar\b|\btrincar\b", str(x)))
  df_frame["Tornar-se_separado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcompor\b|\bdividido\b|\bdividir\b|\bespalhar\b|\bseparar\b", str(x)))
  df_frame["Tornar-se_solto"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bdescolar\b", str(x))))
  df_frame["Tornar-se_visível"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparição\b|\bdespontar\b", str(x))))
  df_frame["Torneio_de_eliminação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcabeça\sde\schave\b|\bchave\b|\bconquistar\svaga\b|\bdisputa\sdo\sterceiro\slugar\b|
\bdisputar\b|\beliminação\b|\beliminar\b|\beliminatórias\b|\bfase\b|\bfinal\b|\bgrupo\b|\bmata-
mata\b|\boitavas\sde\sfinal\b|\boitavas\b|\bpassar\b|\bquartas\sde\sfinal\b|\bquartas\b|\brepescagem\b
|\bseguir\b|\bsemi\b|\bsemifinais\b|\bsistema\seliminatório\b|\btirar\b", str(x)))
  df_frame["Totalizar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcompletar\b|\bno\stotal\b|\btotalizar\b", str(x)))
  df_frame["Trabalhar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcarreira\b|\bdar\sduro\b|\bemprego\b|\btrabalhar\b|\btrabalho\b", str(x)))
  df_frame["Traços_de_personalidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcachorro\b|\bcompetitividade\b|\bcoragem\b|\bcurioso\b|\bdescompromisso\b|\bforte\b
|\bganancioso\b|\bhipócrita\b|\bmodesto\b|\bpersonalidade\b|\bpretensioso\b|\bresponsável\b|\bsensibi
lidade\b|\bteimoso\b|\btímido\b|\bvaidade\b|\bvalente\b", str(x)))
  df_frame["Traduzir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btraduzido\b", str(x))))
  df_frame["Trajar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bcom\b|\broupa\b|\busar\b",
str(x))))
  df_frame["Transferir"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\brepassar\b|\btransferir\b",
str(x))))
  df_frame["Transição_para_uma_qualidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bficar\b", str(x))))
  df_frame["Transição_para_uma_situação"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bficar\b|\bvir\b", str(x))))
```

```
  df_frame["Transição_para_um_estado"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bcrescer\b|\bficar\b|\bvir\b", str(x))))
  df_frame["Transportar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\barrastar\b|\bbuscar\b|\bcarregar\b|\bconduzir\b|\blevar\b|\bmóvel\b|\bpassageiro\b|\bp
egar\b|\bportátil\b|\btaxímetro\b|\btransportar\b|\btransporte\b|\btrazer\b", str(x))))
  df_frame["Transporte"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baéreo\b|\baeroporto\sregional\b|\baeroporto\b|\bagência\sde\saluguel\sde\scarros\b|\b
agência\sde\sviagens\sde\shelicóptero\b|\bbicicletaria\b|\bcadeira\sde\srodas\b|\bestação\sde\smetrô\
b|\bestação\b|\bestacionamento\b|\bitinerário\b|\blinha\b|\blocadora\sde\sveículos\b|\bmarina\b|\bpara
da\b|\bpassagem\b|\bponto\sde\sônibus\b|\bponto\sde\stáxi\b|\bponto\b|\bporto\b|\brodoviária\b|\brodo
viário\b|\brota\b|\bserviço\sde\stransporte\b|\btrânsito\b|\btransporte\spúblico\b|\btransporte\b|\bvoo\b",
str(x))))
  df_frame["Tratar_e_maltratar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\btratar\b", str(x))))
  df_frame["Trocar"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bescambo\b", str(x))))
  df_frame["Turismo_de_atração"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bagroturismo\b|\batração\sturística\b|\batração\b|\batrativo\b|\becoturismo\b|\bexibição\
b|\bingresso\b|\bmeia-entrada\b", str(x))))
  df_frame["Turismo_de_evento"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacontecimento\b|\bevento\b|\bingresso\b|\bmeia-entrada\b", str(x))))
  df_frame["Unidade_calêndrica"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bà\snoite\b|\babril\b|\bagosto\b|\bano\b|\bdécada\b|\bdezembro\b|\bdia\b|\bépoca\b|\bf
eriado\b|\bfim\sde\ssemana\b|\bfim\sde\starde\b|\bfinal\sdo\sdia\b|\bhoje\b|\bhoje\b|\bjaneiro\b|\bjunho
\b|\bmadrugada\b|\bmaio\b|\bmanhã\b|\bnoite\b|\bnoturno\b|\bnovembro\b|\bontem\b|\boutono\b|\bout
ubro\b|\bperíodo\b|\bpernoite\b|\bpôr-do-sol\b|\bquinta-feira\b|\bsábado\b|\bséculo\b|\bsegunda-
feira\b|\bsemana\b|\bsexta-feira\b|\btarde\b|\btemporada\b|\bterça-feira\b|\bverão\b", str(x))))
  df_frame["Usar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baplicar\b|\baproveitar\b|\bempregar\b|\bexploração\b|\breutilizar\b|\busado\b|\busar\b|\
buso\b|\butilizado\b|\butilizar\b", str(x))))
  df_frame["Usar_recurso"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bgastar\b|\busar\b",
str(x))))
  df_frame["Utensílios"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbandeja\b|\bcolher\b|\bespátula\b|\bpanela\b|\bporcelana\b|\btigela\b|\butensílio\b",
str(x))))
  df_frame["Utilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbom\b|\befetivo\b|\bespetacular\b|\besplêndido\b|\bexcelente\b|\bfantástico\b|\bideal\b
|\binefetivo\b|\bmaravilhoso\b|\bótimo\b|\bperfeito\b|\bpreciso\b|\bprestativo\b|\brecurso\b|\bservir\b|\bs
oberbo\b|\bútil\b|\butilidade\b|\bvaler\b|\bvalioso\b|\bvalor\b", str(x))))
  df_frame["Valor_extremo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\binsignificante\b|\bmínimo\b", str(x))))
  df_frame["Veículo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baeronave\b|\bambulância\b|\bautomóvel\b|\bavião\b|\bbalsa\b|\bbarco\b|\bbicicleta\b|\
bbonde\b|\bbuggy\b|\bcaiaque\b|\bcaminhão\spipa\b|\bcaminhão\b|\bcanoa\b|\bcaravela\b|\bcarro\b|\b
carroça\b|\bcarruagem\b|\bcomboio\b|\bconversível\b|\bcruzeiro\b|\bescuna\b|\bhelicóptero\b|\biate\b|\
blimusine\b|\bminivan\b|\bmoto\b|\bnau\b|\bnavio\b|\bônibus\b|\bpatinete\b|\bpedalinho\b|\bpicape\b|\b
quadricicleta\b|\bquadriciclo\b|\bscooter\b|\bsedan\b|\bsubmarino\b|\btanque\b|\btáxi\b|\btobogã\b|\btre
m\b|\btricíclo\b|\bvalsa\b|\bvan\b|\bveículo\b", str(x))))
  df_frame["Veículo_aterrissagem"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baterrissar\b|\bpousar\b", str(x))))
  df_frame["Vencer_o_oponente"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bperder\b",
str(x))))
  df_frame["Veredito"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bconvicção\b", str(x))))
  df_frame["Verificação Verification"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bidentificar\b", str(x))))
  df_frame["Versão_sequência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\báspero\b|\bfinal\b|\bfuncional\b|\bgrosseiro\b|\binicial\b|\boriginal\b|\bpreliminar\b|\bras
cunho\b", str(x))))
  df_frame["Vestuário"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbermuda\b|\bbermudas\b|\bblusa\b|\bbraçadeira\b|\bcalça\b|\bcalçado\b|\bcalção\b|\b
camisa\b|\bcamiseta\b|\bcasaco\b|\bchapéu\b|\bchinelo\b|\bchuteira\b|\bcolete\b|\bfaixa\b|\bjaqueta\b|\
bluva\b|\bmaiô\b|\bmalha\b|\bmeia\b|\bmeião\b|\bmoletom\b|\bnudismo\b|\bpaletó\b|\bquimono\b|\brou
```

```
pa\b|\bsaia\b|\bsamba-
canção\b|\bsapatilha\b|\bsapato\b|\bseda\b|\bshort\b|\bsunga\b|\btecido\b|\btênis\b|\btoalha\b|\btouca\
b|\buniforme\b|\buwagi\b|\bzubon\b", str(x))))
  df_frame["Vetor_tempo"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\ba\spartir\sde\b|\ba\spropósito\b|\bainda\b|\banteriormente\b|\bantes\b|\bapós\b|\bassi
m\spor\sdiante\b|\baté\b|\batrás\b|\bdepois\b|\bdesde\b|\bem\sseguida\b|\benfim\b|\beventualmente\b|
\bfinalmente\b|\bjá\b|\blogo\b|\bna\shora\b|\bpor\súltimo\b", str(x))))
  df_frame["Viagem"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bexcursão\b|\bexpedição\b|\bfazer\sum\stour\b|\bfuga\b|\bitinerante\b|\bjornada\b|\bodi
sseia\b|\bperegrinação\b|\bsafari\b|\btour\b|\bviagem\b|\bviajante\b|\bviajar\b", str(x))))
  df_frame["Vias"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\balameda\b|\bartéria\b|\bauto-
estrada\b|\bavenida\b|\bbeco\ssem\ssaída\b|\bbulevar\b|\bcalçada\b|\bcalçadão\b|\bcaminho\sde\sacesso\b|\bcaminho\b|\bcurso\b|\besquina\b|\bestrada\b|\bfaixa\b|\bferrovia\b|\bfila\b|\bgaleria\b|\blinha\b|\
bpassagem\ssubterrânea\b|\bpercurso\b|\bpista\b|\bponte\b|\bramal\b|\brodovia\b|\brota\b|\brua\b|\btra
jeto\b|\btrilha\b|\btrilho\b|\btúnel\b|\bvereda\b|\bvia\sexpressa\b|\bviaduto\b", str(x))))
  df_frame["Vício"] =  df_frame["lema"].apply(lambda x: len(re.findall(r"\bviciado\b|\bviciado\b", str(x))))
  df_frame["Violência"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bbrutalidade\b|\bselvageria\b|\bviolência\b", str(x))))
  df_frame["Vir_a_existir"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\baparecer\b|\bdesenvolver\b|\bemergir\b|\bfeito\b|\bflorescer\b|\bmaterializar\b|\bnasce
r\b|\brealizar\b|\breaparecer\b", str(x))))
  df_frame["Visitar"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\brevisitar\b|\bvisitação\b|\bvisitante\b|\bvisitar\b", str(x))))
  df_frame["Volubilidade"] =  df_frame["lema"].apply(lambda x:
len(re.findall(r"\bacomodado\somisso\ssilencioso\b", str(x))))
  df_frame.drop('Cenário_do_turismo_estada', axis = 1, inplace=True)
  df_frame['frame_pred_sum']= df_frame.loc[:, "Abundância_distribuída": "Volubilidade"].sum(axis= 1 )
  df_frame= df_frame.query('frame_pred_sum > 0')
  df_frame['frame_pred'] = df_frame.loc[:, 'Abundância_distribuída':  'Volubilidade'].idxmax(axis =1)
  return df_frame
coral_frames = coral_framenet()
coral_frames_f = pd.DataFrame(coral_frames.groupby(['tonal_units'])['frame_pred'].value_counts())
coral_frames_f.columns = ['Frequência']
coral_frames_f.reset_index(inplace=True)
coral_frames_f = coral_frames_f.query('frame_pred != "Cenário_do_turismo_estada"')
fig_final = px.sunburst(coral_frames_f.query('Frequência > 10'), path=['frame_pred'],   color =
'tonal_units', values = 'Frequência',
                color_continuous_scale='BuPu')
fig_final.update_layout(width=700, height=700, margin = dict(t=130, l=50, r=10, b=10), title_text=
f'Principais cenas associadas ao léxico do C-ORAL-ESQ',
                    title_x=0.5, title_y = 0.899, title_font_size= 20)
                    #uniformtext=dict(minsize=13, mode='hide'))
fig_final.add_trace(go.Sunburst(
   insidetextorientation='tangential'
))
fig_final.show()
```

Anexo H – Exemplos e áudios utilizados nesta tese

Pasta com todos os exemplos:

https://drive.google.com/drive/folders/10T05eZ9nqw_p1lSfobha7kJx61R5PnGW?usp=sharing

Exemplo 1 - quebras terminais e não terminais
(áudio bbpubdl01, 25,620 s a 28,620 s):
*PAU: não // tá dando a altura daquele que a Isa marcou lá / né //
Ouça:
https://drive.google.com/file/d/1gfl8sN9qm4sPUGzzffEhVeim6EbtHDJS/view?usp=sharing

Exemplo 2 (bfammn01, 191.880 s a 193.090 s)
aí matou ele //=COM=
Ouça:
https://drive.google.com/file/d/12LlBjT0H9VgeZmTpjBIUILU0yDz7rhZx/view?usp=sharing

Exemplo 3 (bfammn05, 175.620 s a 177.995 s)
eles trouxeram ela pra mim //=COM=
Ouça:
https://drive.google.com/file/d/1p1V7mMvf77ovnHamG3kz0t3i1udzDl_a/view?usp=sharing

Exemplo 4 (áudio bfammn06, de 287.900 s a 177.995 s):
se ele achar que tá /=SCA_r= já na hora /=TOP_r= aí eu ligo pro sior /=CMM_r= sior vai po Otaviano Neves //=CMM_r=
Ouça:
https://drive.google.com/file/d/1tl1j2xQ4aw25CRxQg2RKIMIiUVLoKT0P/view?usp=sharing

Exemplo 5 (bfammn02, 165.440 s a 167.205 s)
não /=CMM_r= nũ acredito nisso não //=CMM_r=
Ouça:

https://drive.google.com/file/d/1M340MmuBXg39pfH79JLe-dAtHXt9NTix/view?usp=sharing

Exemplo 6 (bfammn06, de 372.700 s a 374.606 s):
eu também nũ tenho esse hábito /=COB= nós nũ temo //=COM=
Ouça:
https://drive.google.com/file/d/1sgELLV07_nzWMIVXaL6EDaUO3jQfhPZX/view?usp=sharing

Exemplo 7 (bfammn04, de 12.500 a 15.535 s):

a siora que é mãe /=COB_r= siora sabe /=COB_r= sio’ pega meu filho //=COM_r=
Ouça:
https://drive.google.com/file/d/1JDhR1RnQlZt4ujZ7UYXq2kBxr-3ehm2q/view?usp=sharing

Exemplo 8 (bfammn03 88.420 s a 97.522 s):
ela me fez uma compra lá /=COB_r= comprou presente pr’ ocês todo /=COB_r= pos menino /=COB_r= roupa /=COB_r= e tal /=COB_r= e /=DCT_r= ea nũ me pagou /=CMM_r= também nũ cobrei /=CMM_r= e já perdoei isso há muito tempo /=CMM_r= nem converso mais não //=COM_r=
Ouça:
https://drive.google.com/file/d/15NAyGAaVv9dL3ARF6DzrVCLRN1u2FJEA/view?usp=sharing

Tópico e Apêndice de Tópico (áudio bfammn06, de 569.320 s a 573.968 s):
isso /=TOP= pro nosso negócio /=APT= é um &mun [/2]=SCA= um negócio muito bom //=COM=
Ouça:
https://drive.google.com/file/d/1M0_qvucXTrZ9zwtvx68XpkaCFQ5silP1/view?usp=sharing

Parentético (áudio bfammn01, de 264.580 s a 268.560 s):
no norte de Mina /=TOP= tinha esse [/2]=SCA= antigamente /=PAR= tinha esse tipo de cobra tudo /=COM= né //=AUX=
Ouça:
https://drive.google.com/file/d/1jdQAqNgK_mUn5gOJ8m53c7TQhZePYLBg/view?usp=sharing

Introdutor Locutivo (áudio bfammn05, de 399.180 s a 403.445 s):
falei /=INT= porque /=DCT_r= ela /=SCA_r= gerou você pra mim //=COM_r=
Ouça:
https://drive.google.com/file/d/1nuUUYm56WjGp_nkSillk1CctTmQFi9BX/view?usp=sharing

Apêndice de Comentário (áudio bfammn04, de 339,800 s a 345,333 s)
e ocê /=TOP_r= nessa tranqüilidade /=COB_r= ainda falando de comida /=COB_r= de nũ sei o quê /=COM_r= tal //=APC_r=
Ouça:
https://drive.google.com/file/d/1SWlrxKkmwhRySLOu30-tigsMShCbP0L2/view?usp=sharing

Fático (áudio bfammn03, de 329,740 s a 338,278 s):
eu falei /=INT= não /=PHA_r= e eu /=SCA_r= devo ter a conta ainda /=COB_r= porque eu nũ [/2]=SCA_r= negócio meu /=TOP_r= eu nũ jogo &f [/1]=SCA_r= nada fora /=COB_r= tá tudo guardado //=COM_r=
Ouça:

https://drive.google.com/file/d/1p41Dot5L67wLFzLsDBmL5OdqkbeM9gtC/view?usp=sharing

Incipitário (áudio bfammn02, de 277,680 s a 289,464 s):

papai /=INT= não /=COB_r= é &mov [/1]=SCA_r= poesia moderna /=COB_r= eh meu pai /=ALL_r= e tudo mais /=COB= então /=DCT= o tio Carlos tinha um certo dever de gratidão /=SCA= com o papai /=COB= porque /=DCT= o papai é que /=SCA= foi <estudar direito no Rio> +=COB=

Ouça:
https://drive.google.com/file/d/1IxIi1NBH3imO9TKKe8JcBLQ-iVz1KVhM/view?usp=sharing

Alocutivo (áudio bfamm02, de 62,900 s a 65,819 s):

quando eu falei no nome do papai /=TOP= ele falou /=INT= mas pera aqui /=COM_r= dona Flávia //=ALL_r=
Ouça:

https://drive.google.com/file/d/1CDDLjsn4dE4r164yPZWxoYGcxGpOO5-9/view?usp=sharing

Conativo(áudio bfammn03, de 284,720 s a 289,751 s):

eu falei /=INT= ah /=CNT_r= &Eu =EMP_r= Eustáquio /=ALL_r= larga pa lá isso /=COB_r= eu não /=SCA_r= nũ tô nem /=SCA_r= pensando não //=COM_r=
Ouça:

https://drive.google.com/file/d/1zG6RVGZ88joJKTWXN5NE0KUnb8rfLEHO/view?usp=sharing

Conector discursivo (áudio bfammn05, de 534,520 s a 544,059 s):

/ aí /=DCT= o caminhão virou /=COB= a menina /2=EMP= o capô do carro bateu na /=SCA= fronte dela aqui /=COB= e /=DCT= matou a na hora //=COM=
Ouça:

Ouça:

https://drive.google.com/file/d/1V3fjWQUXBKWH7oTggPmZCmXZ3tf59Dhf/view?usp=sharing

Expressivo (áudio bfamdl01, de 25,640 s a 27,132 s):

*FLA: Nossa /=EXP= aqui tá cheio hoje //=COM=$

Ouça:

https://drive.google.com/file/d/1GesuAJEhrQUj-t96FeepagA8-gvfyWJ3/view?usp=sharing

Tomada de tempo (áudio bfammn01, de 13,220 s a 24,605 s):

&he /=TMT= esse rapaz /=TOP= &he /=TMT= abriu um [/1]=SCA= um claro dentro de uma mata /=COB= pa fazer uma [/1]=SCA= uma plantação /=COB= fazer &u [/1]=SCA= tipo [/1]=EMP= tipo de lavoura /=COM= né //=AUX=

Ouça:

https://drive.google.com/file/d/1wJlzOW6_T6nwD7iWFliyvDuNtJjZlbFV/view?usp=sharing