

UNIVERSIDADE FEDERAL DE MINAS GERAIS
FACULDADE DE LETRAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTUDOS LINGÜÍSTICOS

FILIFE RUBINI CASTANO

BEYOND READABILITY:
A CORPUS-BASED PROPOSAL FOR TEXT DIFFICULTY ANALYSIS

BELO HORIZONTE
2018

FILIPE RUBINI CASTANO

**BEYOND READABILITY:
A CORPUS-BASED PROPOSAL FOR TEXT DIFFICULTY ANALYSIS**

Dissertação apresentada ao Programa de Pós-Graduação em Estudos Linguísticos da Faculdade de Letras da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Linguística Teórica e Descritiva.

Área de Concentração: Linguística Teórica e Descritiva

Linha de Pesquisa: (1D) Estudos Linguísticos baseados em Corpora

Orientadora: Prof.^{fa}. Dr.^a. Heliana Ribeiro de Mello

Belo Horizonte
Faculdade de Letras da UFMG
2018

Ficha catalográfica elaborada pelos Bibliotecários da Biblioteca FALE/UFMG

C346c Castano, Filipe Rubini.
Beyond readability [manuscrito] : a corpus-based proposal for
text difficulty analysis / Filipe Rubini Castano. – 2018.
210p., enc. : il., color., tabs.
Orientadora: Heliana Ribeiro de Mello.
Área de concentração: Linguística Teórica e Descritiva.
Linha de Pesquisa: Estudos Lingüísticos Baseados em Corpora.
Dissertação (mestrado) – Universidade Federal de Minas
Gerais, Faculdade de Letras.
Bibliografia: p. 201-210.
Apêndices: p. 123-200.

1. Linguística – Teses. 2. Linguística de corpus – Teses. 3.
Vocabulário – Teses. 4. Linguística – Metodologia – Teses. 5. I.
Mello, Heliana. II. Universidade Federal de Minas Gerais.
Faculdade de Letras. III. Título.

CDD: 410



UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM ESTUDOS LINGÜÍSTICOS



FOLHA DE APROVAÇÃO


**BEYOND READABILITY:
A Corpus-based Proposal for Text Difficulty Analysis**

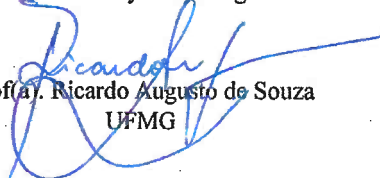
FILIPPE RUBINI CASTANO

Dissertação submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ESTUDOS LINGÜÍSTICOS, como requisito para obtenção do grau de Mestre em ESTUDOS LINGÜÍSTICOS, área de concentração LINGÜÍSTICA TEÓRICA E DESCRITIVA, linha de pesquisa Estudos Linguísticos Baseados em Corpora.

Aprovada em 09 de novembro de 2018, pela banca constituída pelos membros:


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


Prof(a). Vander Paula Viana
University of Stirling


Prof(a). Ricardo Augusto de Souza
UFMG

Belo Horizonte, 9 de novembro de 2018.

ACKNOWLEDGMENTS

After 290 exchanged e-mails, 14 research project drafts, 12 thesis drafts, and fewer actual meetings than she probably would have liked, it is an understatement to say how well my adviser, Prof. Heliana Ribeiro de Mello, has guided me in every step of the way, with blazing fast replies that were always filled with great suggestions; how she has given me sound advice as well as space to not follow it and fail terribly (which corresponds to all the mistakes and errors in this work); and, perhaps most important of all, taught me by example not only to be a better researcher but also a better human.

This project would not have existed without her support, patience, and encouragement, and I offer her—yet again—the 2-gram that is likely to be one of the most frequent and well dispersed in my interactions with her: *thank you!*

I would also like to thank:

- The defense committee members, who shared some of their precious time and experience in evaluating this work: Prof. Bárbara Malveira Orfanó, Prof. Ricardo Augusto de Souza, and Prof. Vander Viana, with special regard to Prof. Ricardo and Prof. Vander, who provided invaluable, helpful criticism;
- The six professors from whom I had the privilege of learning during the course: César Nardelli Cambraia, Larissa Santos Ciríaco, Heliana Ribeiro de Mello, Tommaso Raso, Rui Rothe-Neves, and Ricardo Augusto de Souza. Prof. Ricardo analyzed a version of this project that was about four times as long as mandated. Despite its length, he provided a carefully thought out and insightful assessment that contributed a lot to the project, for which I am very grateful.
- My fellow colleagues in the Master's program: Clarice Fernandes, Gustavo Fonseca, Eduardo Lacerda, Jessica Queiroz, Saulo Santos, Isabelle de Sousa, and João Souza, who were generous and supportive throughout.

*See? What I wrote
about the 2-gram was
true.*

Again, see?

- The professors who were my early inspirations at [UFMG](#): Lúcia Fulgêncio, Heliana Mello, Laura Miccoli, and Adriana Tenuta.
- Profs. Herzila Bastos, Ana Larissa Adorno Marciotto Oliveira, and Magda Veloso, for my 18-month stint as a trainee teacher of English at CENEX – FALE, which proved to be very enriching and influenced this work;
- My dear friends, whose enduring friendship, generosity, and kindness have been a privilege to experience:
 - Mariana Lemos de Aquino; – Mariana Ferreira Quintela;
 - Thaiane Santos Araújo Braga; – Allyson Mendes Rosa;
 - Marcela Diniz Guimarães; – Alisson Carvalho dos Santos;
 - Kent James Polkinghorne; – Rosane Santos Vilaça.
- The best family one could hope for: my parents, Kátia and Sérgio, my brother, Fábio, my grandmother, Sondina, my aunts and uncle Elaine, Márcia and Francklin, all of whom have supported, encouraged, and put up with me all this time.

Lastly, I would like to thank the unsung heroes who have, often for no financial reward, either spent hundreds of hours of their time into making many of the tools that allowed this project to be pursued and this thesis to be written, or helping others to use them: Guido van Rossum for creating the Python programming language¹; Steven Bird, Edward Loper, and Ewan Klein for [NLTK](#)²; Leslie Lamport for \LaTeX ³, Christian Schenk for \MiKTeX ⁴, Jonathan Kew for \XeTeX ; Hàn Thế Thành for [Microtype](#)⁵; André Miede and Ivo Pletikosić for the [ClassicThesis](#)⁶ style; and the [Stack Overflow](#)⁷ community, which helped me in most of the binds a programmer finds oneself in.

¹ <https://www.python.org/>

² <https://www.nltk.org/>

³ <https://www.latex-project.org/>

⁴ <https://miktex.org/>

⁵ <https://ctan.org/pkg/microtype?lang=en>

⁶ <https://bitbucket.org/amiede/classicthesis/>

⁷ <https://stackoverflow.com/>

ABSTRACT

Since the first half of the twentieth century (Flesch, 1948), the task of assessing text difficulty has been primarily tackled by the design and use of readability formulas in many areas: selecting grade level-appropriate books for schoolchildren (Spache, 1953), simplifying dense subjects, such as medical and legal texts (L. M. Baker et al., 1997; Razek et al., 1982), and, in more recent years, assisting writers in making themselves more understandable (*Readable.io* n.d.).

However, there is little empirical demonstration of the validity of readability formulas, as shown for instance in Begeny and Greene (2014), Leroy and Kauchak (2014), Schriver (2000), and Sydes and Hartley (1997), and many of the tools that are currently available for assessing text difficulty, e.g. *ATOS for Text*, *ATOS for Books* (n.d.), Mitsakaki and Truitt (2007), and *Readable.io* (n.d.), depend on those formulas to function. In addition, these tools are quite limited, meant to be used for a specific language, text type, and intended audience.

In this work, we develop a corpus linguistics-based, lexicon-oriented approach to propose a Text Difficulty Scale (TDS) which, conversely to previous efforts, can be adapted for texts of virtually any language, including those that use non-Latin writing systems. To that end, we have used sounder statistical measurements, such as deviation of proportions (DP) (Gries, 2008, 2010); included 2-grams and 3-grams as sources of numerous yet often disregarded idioms and phrasemes (Bu et al., 2011, p. 3); and built a 60+ million token collection of Wikipedia articles in English for demonstration purposes. Furthermore, we have made our work available, free and open-source, as [a set of Jupyter Notebooks in the Python programming language](#).

We argue that our proposal not only offers a much-needed flexible measurement of text difficulty, in particular for teachers and students of foreign languages, but also that it could be useful for researchers in cognitive linguistics and psycholinguistics, editors, writers, and children acquiring their first language.

RESUMO

Desde a primeira metade do século 20 (Flesch, 1948), a tarefa de avaliar a dificuldade de textos tem sido primariamente enfrentada através do design e uso de fórmulas de legibilidade (*readability formulas*) em diversas áreas: a seleção de livros para crianças em determinadas séries escolares (Spache, 1953), a simplificação de assuntos complexos, como textos de medicina e de direito (L. M. Baker et al., 1997; Razek et al., 1982) e, em anos recentes, auxiliar escritores a se tornarem mais inteligíveis (*Readable.io* s.d.).

Contudo, há pouca demonstração empírica da validade de fórmulas de legibilidade, como evidenciado, por exemplo, em Bengeny e Greene (2014), Leroy e Kauchak (2014), Schriver (2000) e Sydes e Hartley (1997), e muitas das ferramentas que estão disponíveis para a avaliação de dificuldade de texto, por exemplo Mitsakaki e Troutt (2007), dependem dessas fórmulas para funcionar. Além disso, essas ferramentas são bastante limitadas, feitas para serem usadas com uma língua, tipo de texto, e público específicos.

Neste trabalho, desenvolvemos uma abordagem baseada em corpus e focada no léxico, para propor uma *Escala de Dificuldade de Texto* (EDT), a qual, ao contrário de abordagens anteriores, é adaptável a textos em praticamente qualquer língua, incluindo as que utilizam sistemas de escrita não latinos. Para alcançar esse objetivo, utilizamos medidas estatísticas mais sólidas, tais como o *desvio de proporções* (DP) (Gries, 2008, 2010); incluímos *2-grams* e *3-grams* como fontes de expressões numerosas e frequentemente negligenciadas (Bu et al., 2011, p. 3); e construímos uma coleção de textos de mais de 60 milhões de tokens de artigos da Wikipedia em inglês, para demonstração. Ademais, tornamos nosso trabalho de código livre disponível gratuitamente, como um conjunto de *Jupyter Notebooks* escritos na língua de programação Python.

Argumentamos que nossa proposta não somente oferece uma medida flexível e muito necessária de dificuldade de textos, especialmente no que tange a professores e alunos de línguas estrangeiras, mas que também poderia ser útil para pesquisadores em linguística cognitiva e psicolinguística, editores, escritores, e crianças em processo de aquisição de sua primeira língua.

ON STYLE

POS LIN resolution 03/2013 (UFMG, 2013), later reedited on April 2018, states that “the appropriate style guides” should be followed when writing a dissertation or thesis in a foreign language.

However, there is at least a dozen different style guides for publications in English-speaking countries, and each university either has its own specifications for the formatting of theses or recommend their students to choose one and stick to it, as is the case with MIT (Massachusetts Institute of Technology, n.d.).

We have chosen an author-year, Harvard-esque citation style that includes publication dates and can be parenthetical (Meade and Smith, 1991) or not, e.g. Meade and Smith (1991).

In the electronic (.pdf) version of this thesis, sections of the text in blue, green, and red work as hyperlinks, so it is possible to jump easily to the desired section (for instance, to check the bibliography). The bibliography, in turn, lists the pages of the thesis where the citations appeared. Acronyms work in a similar way – for instance, clicking on COCA takes you to the list of acronyms.

As to formatting, margin sizes, and typography, we have followed Bringhurst (2004)’s seminal recommendations as adapted by André Miede and Ivo Pletikosić’s for the *ClassicThesis* style.

This thesis will make use of margin notes which expand on or clarify the subjects treated on the body of the thesis.

This is an example of a margin note.

CONTENTS

1	INTRODUCTION	1
1 THEORETICAL UNDERPINNINGS		
2	THE MANY FACETS OF DIFFICULTY	5
2.1	Difficulty as a concept	5
2.2	Desirable difficulty	6
2.3	Actual and perceived difficulty	6
2.4	Relative text difficulty	7
2.5	Non-relative text difficulty	8
2.6	What about grammar?	9
2.7	Vocabulary and proficiency	10
3	READABILITY	11
3.1	Type A: Orthography-based formulas	11
3.2	Type B: Familiarity-based formulas	12
3.3	Effectiveness of readability formulas	12
3.3.1	Correlations to the Oral Reading Fluency test	13
3.3.2	Correlations to cloze scores	13
3.4	Summary	15
4	FREQUENCY AND DISPERSION	17
4.1	Word familiarity and word frequency	18
4.2	Scientific evidence	18
4.2.1	Kandula et al. (2010): lexical simplification	18
4.2.2	Leroy, Kauchak, and Mouradi (2013): lexical simplification	19
4.2.3	Leroy and Kauchak (2014): word frequency influences difficulty	19
4.2.4	Martinez-Gómez and Aizawa (2013): a Bayesian approach	21
4.2.5	Summary	21
4.3	Word dispersion	23
4.3.1	Word frequency vs. word dispersion	23
4.3.2	Scientific evidence	24
5	MULTIWORD EXPRESSIONS	27
5.1	Martinez and V. A. Murphy (2011)	28

5.2	Summary	29
6	POLYSEMY	31
6.1	Defining polysemy	31
6.2	Polysemy and frequency	32
6.3	Counting polysemy	33
6.4	Accounting for polysemy	33
6.5	Summary	34
II METHODOLOGY		
7	EXISTING APPROACHES	37
7.1	Read-X	37
7.2	Readable.io	37
7.3	WordandPhrase.info	37
7.4	ATOS	38
7.5	DeLite	38
7.6	Our approach	39
7.6.1	User requirements	39
8	THE TEXT COLLECTION	41
8.1	Size	41
8.2	Text source	42
8.3	Processing the dump file	43
8.4	Going big or going deep?	43
8.5	Text file structure	44
9	CHOOSING A LEXICAL UNIT	47
9.1	Word families	48
9.2	Lemmas	50
9.3	Alphabetical types	51
9.4	Types	51
9.5	Choosing a lexical unit	53
9.5.1	What is a word?	53
9.5.2	The learner's perspective	55
9.5.3	Tokenization and our definition of word	57
9.6	Summary	59
10	FROM TEXT TO DATA	61
10.1	Experiments on n -gram coverage	61
10.2	Converting to lowercase	63
10.3	From text to tokens	63
10.4	N -gram generation	66
10.5	The database	66
10.6	Final Wikipedia data information	68

10.7	Summary	68
III THE TEXT DIFFICULTY SCALE		
11	DIFFICULTY AT THE WORD LEVEL	73
11.1	Frequency, dispersion, and the data	73
11.2	Measuring dispersion	77
11.3	Calculation of DP	80
11.4	The distribution of dispersion in the WP data	81
12	DIFFICULTY AT THE TEXT LEVEL	83
12.1	Unique-to-total token ratio	83
12.2	Occurrences and unique tokens	85
12.2.1	2-grams and 3-grams	88
12.3	Our proposal for the TDV	92
12.4	Other factors in text difficulty	92
12.5	Summary	95
13	EXPLORING TEXT DIFFICULTY VALUES	97
13.1	Spoken—academic English	97
13.1.1	Spoken data	98
13.1.2	Academic data	98
13.1.3	Calculating Text Difficulty Values	99
13.1.4	Analysis results	99
13.2	Further demonstrations	100
13.2.1	The 475 random Wikipedia articles	100
13.3	Spoken—academic Portuguese	102
13.3.1	The spoken data	103
13.3.2	The academic data	104
13.4	Analysis results	105
13.5	Summary	107
14	THE DIFFICULTY HIGHLIGHTER	109
14.1	Design considerations	109
14.2	<i>Johann Sebastian Bach</i> in five languages	110
14.2.1	English	111
14.2.2	Portuguese, Japanese, Hebrew, and Persian	113
15	FINAL REMARKS	119
IV APPENDIX		
A	EXTRACTING WIKIPEDIA DUMP FILES	125
A.1	Working with the Wikipedia Dump	125

A.1.1	Extracting the Wikipedia dump into separate files	125
A.1.2	Choosing text files randomly	127
A.2	Text data	128
B	THE TEXT DIFFICULTY ANALYZER CODE	129
B.1	Automatic Wikipedia extractor	129
B.2	The corpus builder	132
B.3	The Text Difficulty Analyzer + Difficulty Highlighter	147
C	ASSEMBLING THE ACADEMIC AND SPOKEN CORPORA	171
C.1	Obtaining TDVs from the SketchEngine wordlists	171
C.2	English	172
C.2.1	The academic text corpus	172
C.2.2	The spoken text corpus	173
C.3	Example of the cleaning results for Portuguese	175
D	DIFFICULTY HIGHLIGHTER TRANSLATIONS	177
E	TOKEN LISTS	183
E.1	English	183
E.2	Portuguese	186
E.3	Japanese	190
E.4	Hebrew	193
E.5	Persian	197
	BIBLIOGRAPHY	200

LIST OF FIGURES

- Figure 4.1 Average actual difficulty for words grouped by word frequency of occurrence, from Leroy and Kauchak (2014, e171). 20
- Figure 6.1 Polysemy and frequency. Reproduced from Zipf (1949). 32
- Figure 10.1 Concept map of the process of data extraction. 64
- Figure 10.2 Two scenarios of n -gram extraction, with (top) and without (bottom) sentence tokenization. Inappropriate n -grams are shown in red, with strings inside single quotation marks. 65
- Figure 12.1 Histograms of the number of unique 1-grams and total occurrences of 1-grams (y axis) and their ranges of DP values (x axis) for the Wikipedia (WP) article *Miscegenation*. 86
- Figure 12.2 Histograms of the number of unique 2-grams and total occurrences of 2-grams (y axis) and their ranges of DP values (x axis) for the WP article *Miscegenation*. 89
- Figure 12.3 Histograms of the number of unique 3-grams and total occurrences of 3-grams (y axis) and their ranges of DP values (x axis) for the WP article *Miscegenation*. 90
- Figure 14.1 *Johann Sebastian Bach* Portuguese WP article highlighted for difficulty. 114
- Figure 14.2 *Johann Sebastian Bach* Japanese WP article highlighted for difficulty. 115
- Figure 14.3 *Johann Sebastian Bach* Hebrew WP article highlighted for difficulty. 116
- Figure 14.4 *Johann Sebastian Bach* Persian WP article highlighted for difficulty. 117

Figure D.1	<i>Johann Sebastian Bach</i> Portuguese WP article translated into English. 178
Figure D.2	<i>Johann Sebastian Bach</i> Japanese WP article translated into English. 179
Figure D.3	<i>Johann Sebastian Bach</i> Persian WP article translated into English. 180
Figure D.4	<i>Johann Sebastian Bach</i> Hebrew WP article translated into English. 181

LIST OF TABLES

Table 4.1	Breakdown of lemma composition in the Oxford English Corpus (OEC) according to the top x most frequent lemmas. Adapted from https://goo.gl/CvfPY1 . 17
Table 9.1	Occurrences of PLAY word forms in the Corpus of Contemporary American English (COCA). 52
Table 10.1	Overall information on the Wikipedia data in the first run of experiments with 129,000 articles. 61
Table 10.2	Overall statistics for the final Wikipedia 124,000-article text collection. 69
Table 11.1	Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs equally throughout the corpus. 79
Table 11.2	Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs in only one part. 80
Table 11.3	Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs only in the largest part. 80
Table 11.4	The number of unique tokens in each range of DP values in the WP data. 81

Table 12.1	Analysis of 475 WP articles according to their mean and median article lengths, average unique-to-total ratio, and DP.	91
Table 13.1	Results of the spoken–academic text comparison.	100
Table 13.2	Difference in text difficulty values (TDVs) after ascribing the value of 1.000 to tokens not found in the English data.	102
Table 13.3	Academic Portuguese data statistics, with the number of tokens estimated by SketchEngine.	105
Table 13.4	Results of the spoken–academic text comparison for English and Portuguese.	106
Table 13.5	Difference in TDVs after ascribing the value of 1.000 to tokens not found in the Portuguese data.	107
Table 14.1	Overall statistics for the Wikipedia text data in English, Portuguese, Japanese, Hebrew, and Persian.	111
Table C.1	Santa Barbara Corpus of Spoken American English (SBCSAE) transcription file before and after cleaning.	174
Table C.2	C-ORAL-BRASIL (COB) transcription file before and after cleaning.	176
Table E.1	Top 100 English 1-grams	183
Table E.2	Top 100 Portuguese 1-grams	186
Table E.3	Top 100 Japanese 1-grams	190
Table E.4	Top 100 Hebrew 1-grams	193
Table E.5	Top 100 Persian 1-grams	197

LISTINGS

Listing 10.1	Example of MongoDB document for the token <i>serendipitous</i>	67
Listing 10.2	Different queries for the token database	67
Listing A.1	Extracting subdirectories of WP dump into text files	125
Listing A.2	Choosing text files randomly	127
Listing B.1	Automatic Wikipedia extractor	129

Listing B.2	The corpus builder	132
Listing B.3	The Text Difficulty Analyzer + Difficulty Highlighter	147
Listing C.1	Obtaining DP and TDVs from SketchEngine wordlists	171
Listing C.2	Choosing academic article files randomly	172
Listing C.3	Cleaning the Santa Barbara corpus files	173

ACRONYMS

BNC	British National Corpus
CD	contextual diversity
COB	C-ORAL-BRASIL
COCA	Corpus of Contemporary American English
DP	deviation of proportions
MWE	multiword expression
NLTK	Natural Language Toolkit
OEC	Oxford English Corpus
OED	Oxford English Dictionary
ORF	Oral Reading Fluency
RF	relative frequency
SBCSAE	Santa Barbara Corpus of Spoken American English
TDS	Text Difficulty Scale
TDV	Text Difficulty Value
UFMG	Universidade Federal de Minas Gerais
WD	word dispersion
WF	word frequency
WP	Wikipedia

INTRODUCTION

Language teachers are often tasked with selecting authentic or appropriate pedagogical materials for their students (Okamoto, 2015, p. 9). They either rely on “readers”, usually simplifications of famous literary works for specific grade levels (Davanzo, 2016), or on their own instincts as teachers, facing the risk of under- or overestimating their students’ reading abilities.

This work originated from the intention to aid teachers and learners of languages in selecting more appropriate texts, i.e. texts that are more appropriate difficulty-wise for the proficiency level of the reader.

The concept of assessing text difficulty is not new; it has been around since at least the early twentieth century, one such example being Flesch (1948). The first attempts to assess difficulty were readability formulas, which are used to this day. However, as we will show in chapter 3, despite the great number of readability formulas, they present little evidence of utility.

Therefore, we need a tool that is able to assess difficulty in a more reliable manner by taking into consideration the available scientific evidence, in particular the linguistics-related evidence. Most readability formulas are focused on children learning their first language (Benjamin, 2012, p. 83); instead, we mean to focus on teachers and learners of second or foreign languages.

We argue that such an endeavor is justifiable by its potential, among other things, to:

1. Restate the importance of vocabulary in language learning and second language acquisition, areas where grammar is often one of the main targets of interest (Cook, 2008, p. 6);
2. Aid learners in choosing reading materials appropriate for their current proficiency level, as well as assisting teachers and language professionals in that regard, by reducing or eliminating much of the work in guessing whether a text is difficult or not;
3. Inform the formulation of language courses, learning materials, and vocabulary books;

4. Make the comparative analysis of texts easier wherever word frequency, dispersion, and other vocabulary-related aspects are concerned.

Computer programs that attempt to assess text difficulty do exist; however, they present at best limited applicability. We outline and compare them to our own approach in chapter 7, p. 37. With the ultimate goal of constructing such a program in mind, this work is divided into three parts:

- In Part I, we examine the theoretical and empirical research in terms of the potential connections between corpus measurements – such as word frequency, word dispersion, and word familiarity – and text difficulty;
- In Part II, we apply what can be learned from such a research framework into the development of a flexible, hopefully improved methodology for text difficulty assessment;
- Finally, in Part III, we demonstrate our proposal for text difficulty assessment on two different languages, as well as our Difficulty Highlighter application on five different languages.

Part I

THEORETICAL UNDERPINNINGS

The theoretical and empirical background to the notions of readability, corpus measurements such as word frequency and word dispersion, and their potential influence on text difficulty.

THE MANY FACETS OF DIFFICULTY

Everyone is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid.

— Unknown author;
often wrongfully attributed to Albert Einstein

One of the most deceptively simple concepts to define, in linguistics or any other field, is **difficulty**. It is easy to overlook or take for granted; for a work that relies so much on the concept, we must spend some time exploring its properties. In this section, we will discuss the concept of difficulty in more general (common sense) terms; what difficulty means in learning; and which linguistic factors, if any, could influence the difficulty level of a text.

2.1 DIFFICULTY AS A CONCEPT

The dictionary definition of difficulty is straightforward: “the state or condition of being difficult; (...) viz. hard to accomplish, deal with, or understand (Press, n.d.)”.

We must, however, remember that difficulty *can* be **relative**. A fish has little difficulty to swim, yet will face insurmountable difficulty when attempting to climb a tree. Thus, each individual has a different skill set or background, which influences how hard something is for that particular individual.

The relativity of difficulty presupposes that certain **underlying factors can influence difficulty**. In the tree-climbing fish example, the shape, size, and even instinct of the animal could influence its difficulty for climbing trees.

The dictionary definition, however, conveys a binary quality to difficulty: either something is difficult or not. However, acknowledging the existence of factors that influence difficulty (i.e. increase or decrease it) allows for building a **scale of difficulty**, e.g.

This chapter is not a thorough visitation of the subject of difficulty. We will focus on the aspects of difficulty that could help in creating a text difficulty analyzer.

ranging from *easy* to *difficult*. Thus scales or degrees of difficulty depend on the factors influencing difficulty (the background of the task and of its actors etc.) as well as on how the scale designer represents and assigns a weight to each of those factors.

Nevertheless, difficulty can also be independent of skill and/or background, i.e. **non-relative**. For two able-bodied identical twins with similar fitness backgrounds, climbing a step in a flight of stairs is much less difficult than climbing Mount Everest.

2.2 DESIRABLE DIFFICULTY

Desirable difficulty is a concept in cognitive psychology. It basically proposes that, when designing the training for a given task, introducing difficulties for the learner increases retention and success in that task (Bjork, 1994, p. 189). The assumption behind desirable difficulty is that the more extensive the processing, and consequently the deeper the cognitive strategy, the better is the learning. Desirable difficulty has not always been found to correlate significantly with higher performance, but it would be nonintuitive to claim that by providing students solely with easy tasks (in our project's case, unchallenging texts), their learning will improve.

We are of the opinion that there must be a balance: the desirable spot in text difficulty is somewhere between the overbearingly difficult and the tediously simple. Especially for students of languages, they must be “injected” periodically with a healthy dose of new words and expressions. In order to do this, teachers and students need to be able to clearly separate what is too difficult from what is too easy.

2.3 ACTUAL AND PERCEIVED DIFFICULTY

In addition to being influenced by relative and non-relative factors, there are other ways we can look at difficulty. We can talk of **perceived difficulty**, i.e. the impressions of an individual regarding how difficult a given task is. **Actual difficulty**, on the other hand, measures difficulty through tests to ascertain objectively how well the subject fared in the task, as in Leroy and Kauchak (2014).

For the effect of difficulty in performance, see Deslauriers et al. (2012), p. 82, for some examples of weak or insignificant correlations, and studies like Hughes et al. (2013) and Pyc and Rawson (2009) for significant correlations.

Testing, we argue, is the gold standard to scientifically gauge actual difficulty, considering that it excludes the possibility that one may over or underestimate their comprehension of the text, as one study (Martinez and V. A. Murphy, 2011) we discuss in section 5.1, p. 28, shows; yet, the psychological element of difficulty cannot be completely ignored.

Having established the different facets of difficulty (relative, non-relative, perceived, and actual) we must approach the subject of our work, **text difficulty**, investigating it from these four perspectives.

2.4 RELATIVE TEXT DIFFICULTY

Relative text difficulty is influenced by the social, economic, and linguistic background of the reader. For instance: a native speaker of Portuguese learning English as a foreign language will probably not find much trouble when encountering the word *gingivitis*, as it is similar to a Portuguese word of same meaning, *gengivite*, thus potentially accelerating the learning of the word. An English native speaker would be more likely to know the expression *gum swelling*, as observed in Maylath (1997).

Similarly, a white kid growing up in an upper-class family in Manhattan may have to rely on their context-awareness skills in order to understand some of the African-American English spoken in inner city ghettos (Bailin and Grafstein, 2001, p. 288).

Career choices also influence difficulty. For instance, a physicist will likely understand a paper on quantum tunneling¹ with less difficulty than the average person.

We acknowledge the importance of social factors in influencing difficulty. However, this work does not intend to dwell on them, for they are too numerous to control: imagine trying to adjust for factors like native tongue, gender, age, area of birth, ethnicity, income level, education level, number and subjects of books read, career history, travels undertaken, religion, and a myriad of other life experiences, in order to gauge difficulty of a text for a person.

Lingo can even be used to increase difficulty, as in the case of Polari, 19th century slang shared by homosexual individuals in Britain as a disguise, as homosexuality was illegal at the time (P. Baker, 2003).

¹ Quantum tunneling could create new Big Bangs in the vacuum in the far future, giving rise to new universes (Carroll and Chen, 2004).

It is simply not feasible to go on this route if you consider the diversity of humankind: people can be many things at the same time, and it may not always be that a person who belongs in a group (say, Jews, African-Americans, atheists etc.) will necessarily know the lingo or jargon of that group.

On the other hand, one crucial facet of relative difficulty, in particular for language learners, is vocabulary size (I. Nation, 2006; P. Nation, 1997; P. Nation and Coady, 1988). If a student of English as a foreign language knows only a few thousand words in their vocabulary – many of which we can assume are the most frequent words – they will probably struggle with some types of texts (contemporary and 20th century novels, academic and newspaper articles) and have an easier time with others (sitcoms, talk shows, informal conversation, etc.).

The most conservative study we could find for English vocabulary size (Goulden et al., 1990) arrived at a “native-like” vocabulary consisting of 17,000 words, whereas the other side of the spectrum found figures as large as 216,000 words (Diller, 1978), which of course depends on what is defined as a word.

2.5 NON-RELATIVE TEXT DIFFICULTY

It is a foregone conclusion that, independently of language background, socioeconomic status, gender, etc., it is of paramount importance to learn the meanings of the building blocks of texts – words and expressions – either directly (through, for instance, word definitions) or indirectly (through context or other means), to understand them. If one’s vocabulary is limited, understanding a text becomes too difficult or outright impossible.

Having a diverse **vocabulary** (in this work used to mean the set of words an individual knows, in opposition to **lexicon**, which comprises all the words and expressions in a language) is not only important for children learning to speak and read, but also to those studying a second or foreign language (P. Nation and Coady, 1988). For the latter, especially as adults, the effort can be daunting, because the lexicon of any language has dozens of thousands of words and expressions (emphasis added):

Knowing words is the key to understanding and being understood. **The bulk of learning a new language consists of learning new words.** Grammatical knowledge does not make for great proficiency in a language (Vermeer (1992), p. 147)

Students of foreign languages face a particular struggle in learning new vocabulary. Although learning materials present abundant instruction on pronunciation, conversational strategies, grammar rules and numerous grammatical exercises, they do not usually present a sufficiently large or useful set of words, as demonstrated by O’Loughlin (2012). In fact, it is impractical to try to cram a language’s lexicon into didactic textbooks; not only the books would become unwieldy, but there is also a limitation on how much of English’s lexical diversity one can include in a language course, for reasons of time, pedagogical principles, etc.

2.6 WHAT ABOUT GRAMMAR?

As shown previously, the often understated importance of vocabulary, and the amount of effort required to acquire it, is among the reasons why, in this project, we will focus on the lexicon rather than on grammar as a measure of difficulty.

Naturally, vocabulary deficiency is not the only factor in text difficulty. The target language’s grammar, and the errors that are bound to occur when one learns it, do shed an important light on the process of language acquisition; some grammatical constructions seem to be more difficult to process and learn than others, and there is a multitude of research on different languages and grammatical domains, such as argument realization and morphology (Boerma et al., 2017; Souza and Mello, 2007).

However, attempting to integrate different linguistic factors (for instance, the complexity of the morphosyntactic system – whether it has case, gender, number, etc., and the properties of these features; number of irregular verb forms; number and specific properties of verbal conjugation; and so on) into a unified “theory of difficulty” that would predict text difficulty for a wide range of languages would likely be unfruitful, as languages can be (and often are) very different from each other. There is an “amazing diversity of linguistic structures”, as persuasively argued by Evans and Levinson (2009, p. 445).

Thus, focusing on the lexicon allows a single framework to be applied on many languages – considering that one would be hard pressed to find a language without some sort of lexicon – making unnecessary the inclusion of specific details of the grammar of in-

dividual languages (as well as the assessment of learners' degree of mastery of said details). The only requirement would be obtaining texts on the target language and finding or training a tokenizer for that language.

2.7 VOCABULARY AND PROFICIENCY

The difficulty in acquiring new vocabulary can be remedied by reading, in particular **extensive reading**, as demonstrated in the literature. According to P. Nation (1997), the benefits of expanding one's own vocabulary through extensive reading are not limited to just how much one can understand of the language:

For studies on reading and its effects on language ability, see Saragi et al. (1978), Renandya and Jacobs (2016), Pigada and Schmitt (2006), and P. Nation and Coady (1988).

experimental studies have shown that not only is there improvement in reading, but that there are improvements in a range of language uses and areas of language knowledge (p. 16).

By this principle, we could make the case that the more texts read and the more words an individual has learned², the better for their overall proficiency level.

However, it is in theory possible to learn a language more efficiently, since some words are more useful than others. One would probably not say that the noun *aardvark* is, in general, more useful than the verb *eat* for a language student. The problem becomes: how do we know *which* words are useful and which are not? And, by extension, which texts contain the most useful words for the language student?

² In this work, knowing at least one definition of a word will be referred to as “learn a word”, “acquire” or “know a word”, etc. interchangeably. By those terms, we are referring to vocabulary acquisition by both children and adults, as well as to first and second language vocabulary acquisition.

READABILITY

Some of the earliest efforts in grading text difficulty were readability formulas. **Readability** can be defined as the measure of how easy a given text is to understand. Therefore, the more readable a text is, the easier it is. Traditional readability research concerns itself with the design of suitable **readability formulas** to ascertain how readable a given text is (Benjamin, 2012; Meade and Smith, 1991).

Readability formulas have been the *de facto* standard for gauging relative difficulty of texts. Their number is probably in the hundreds (Benjamin, 2012, p. 63), and their use is widespread. Some of their applications are laid out by Begeny and Greene (2014, p. 199):

- simplify texts that are perceived as hard to understand due to their subject matter, such as accounting textbooks (Razek et al., 1982), surgical consent forms (Grundner, 1980), or medical texts (L. M. Baker et al., 1997);
- select, simplify, and grade texts for young native speakers of English in school years (Spache, 1953);
- grade exams and entrance forms for the US Army (Sticht, 1973).

Formulas come in many shapes, sizes, and intended targets (Begeny and Greene, 2014). We will now discuss the two main types of readability formulas, which we have termed in this work **Type A** and **Type B**.

3.1 TYPE A: ORTHOGRAPHY-BASED FORMULAS

This type, which includes the influential Flesch formula (Flesch, 1948), measures the number of syllables in words and how many words a sentence contains. In other words, this type of formula assumes that both long words and word-filled sentences have an impact on reading comprehension.

Using these criteria to establish readability is a questionable decision, argue Bailin and Grafstein (2001), p. 289:

First of all, there appear to be a significant number of instances where mono or bisyllabic words are more esoteric, more unfamiliar, than longer polysyllabic terms. Consider, for example, the short, monosyllabic *curr*, and the longer, morphologically complex, *reinventing*. The number of readers, even children, who know the latter term is quite likely greater than those who know the former. Is the word *aardvark* more familiar than *unemployment*? One would hardly think so. But again, the reading formulas in question would treat the latter as contributing more to the complexity of the text than the former.

3.2 TYPE B: FAMILIARITY-BASED FORMULAS

On the other hand, **Type B** readability formulas, as the Dale-Chall (Chall and Dale, 1948) and its reincarnation 47 years later (Chall and Dale, 1995), take into consideration **word familiarity**. In the case of Dale-Chall, if there are many words in a text that do not belong to a list of the 3,000 most familiar words, the *less readable* it is. We will discuss word familiarity and its related concepts, word frequency and word dispersion, in chapter 4.

3.3 EFFECTIVENESS OF READABILITY FORMULAS

Despite their variety and extensive use, “the validity of readability formulas is inconclusive in the scientific literature” (Begeny and Greene, 2014, p. 201); “there is little evidence that readability formula outcomes relate to text understanding” (Leroy and Kauchak, 2014, p. 169); using readability formulas to perform revision of text has been shown to be unsuccessful in terms of improvements in comprehension (Duffy and Kabance, 1982); their limitations can make them quite misleading (Pichert and Elam, 1985), and a critical review (Schriver, 2000, p. 140) states that there are only two upsides to their use:

the formulas serve to remind people who would otherwise be unaware of issues of readability to consider them (...) [and] have served the very useful function of igniting debate among (...) researchers.

Before taking these researchers' statements as fact, let us explore the evidence on readability formula effectiveness in a little more detail.

3.3.1 *Correlations to the Oral Reading Fluency test*

What is most surprising in regards to the empirical evidence on readability is that the majority of researchers in the field did not seem to correlate formulas to *actual* difficulty or even perceived difficulty; instead, they often made correlations to the Oral Reading Fluency (ORF) test, which simply assesses how well children were able to read texts aloud¹ (Begeny and Greene, 2014). Logically, being able to do so does not entail an understanding of the words being spoken, especially if the subjects are children.

Even correlating readability formulas to such an inappropriate measurement of difficulty yielded little to no result in several studies, as stated in Begeny and Greene (2014, pp. 201–203), with only the Dale-Chall readability formula (Type B, word frequency-based) ultimately showing a correlation to ORF performance (Begeny and Greene, 2014, p. 209).

A list of the ORF-related readability studies is available in Begeny and Greene (2014), p. 201.

3.3.2 *Correlations to cloze scores*

Let us now examine the available evidence – which is in very short supply – when it comes to readability formulas and *actual* difficulty. The most practical way to assess the level of reading comprehension – and, by consequence, difficulty – seems to be **cloze tests**, which are constructed by a procedure similar to this:

replacing every fifth word in a passage with an underlined blank of a standard length. Students who have not read the intact passage are asked to write in each blank the word they think was deleted, and their responses are scored correct only when they exactly match the word deleted (Bormuth (1971), p. 13).

¹ <https://goo.gl/uhjfyD>. Accessed on 11/01/2018.

The rationale is that cloze tests share the same underlying principles of traditional comprehension tests. However, assigning as a correct response only the exact match of the original text seems a bit restrictive. For instance:

“I saw a man lay his jacket on a puddle for a woman crossing the street. I thought that was very _____”²

In this passage, there are several options for the blank: *romantic*, *chivalrous*, *gallant*, and depending on the test taker’s opinion, *cheesy* or even *foolish*. All of these options seem valid, and they could reveal a good amount of information on the subject’s vocabulary. If one limits the correct answer to only one alternative – say, *romantic* – the results of the test have at best limited value.

A 2009 study on 25 African-American men’s understanding of prostate cancer information found that “correlations between Flesch–Kincaid readability and Cloze comprehension scores were not significant” (Friedman et al., 2009, p. 454), Flesch-Kincaid being one example of Type A formulas.

Some readability formulas, such as Dale-Chall, have been considered “valid” empirically by Benjamin (2012, p. 81). A closer look at this statement reveals that this validity is based on performance on these limited-type cloze tests, especially those developed in Bormuth (1971) – a U.S. government Office of Education report – which included several other factors, for instance students’ opinions of the passages and whether they wanted to continue reading (i.e. an assumption that if an elementary student shows interest on a passage, the passage must have been readable). Not to mention that Bormuth’s “evidence” focuses mainly on the suitability of textbooks and pedagogical materials.

Benjamin (2012, p. 66) further states that the Dale-Chall formula, in addition to having been found valid on cloze tests, “was also successfully validated by comparing predicted difficulty levels with various standardized reading tests”. No sources are given for those standardized tests, making us suppose that these may well be the ORF tests.

² Adapted from <https://goo.gl/2Njxny>. Accessed on 01/06/2018.

3.4 SUMMARY

We have thus far outlined the following negative aspects of readability research and readability formulas:

- Using criteria such as the length of words or the number of words in a sentence is not the best methodology to assess difficulty, as there is plenty of short words (e.g. *gouge, bias, err*) that could be mistakenly thought of less difficult than longer words (e.g. *unemployment, teasing, amazing*).
- Most of the research on readability shows their limited validity, with formulas failing to correlate even with the ORF tests;
- The results of actual difficulty tests e.g. cloze tests, depending on how they are designed, can be either limiting or misleading.

Only in the mid 2010's we see studies, such as Leroy and Kauchak (2014), gauging actual difficulty of specific components of readability formulas, such as word frequency. In the next chapters, we will look into those components separately.

FREQUENCY AND DISPERSION

Human language, despite its immense potential for creativity and innovation, is often very predictable and formulaic, especially in terms of the lexicon (Griffiths, 2011). One analysis of the OEC¹ shows that a very small amount of lemmas account for a great deal of the content in most text corpora, as shown in table 4.1.

Table 4.1: Breakdown of lemma composition in the OEC according to the top x most frequent lemmas. Adapted from <https://goo.gl/CvfPY1>.

Top x most frequent lemmas	% of content in OEC	Example lemmas
10	25%	THE, OF, AND, TO, THAT, HAVE
100	50%	FROM, BECAUSE, GO, ME, OUR, WELL, WAY
1000	75%	GIRL, WIN, DECIDE, HUGE, DIFFICULT, SERIES
7000	90%	TACKLE, PEAK, CRUDE, PURELY, DUDE, MODEST
50,000	95%	SABOTEUR, AUTOCRACY, CALYX, CONFORMIST
>1,000,000	99%	LAGGARDLY, ENDOBENTHIC, POMOLOGICAL

Far from an equal distribution, there is a small set of words that are highly frequent, in opposition to a very large set of words that are comparatively uncommon, something that was noticed in the mid-twentieth century (Zipf, 1949).

Word frequency, a statistical measure applied to a corpus that counts the occurrences of each word either in relation to the entire corpus (**absolute frequency**, as in *347 occurrences*) or in relation to the other words in the corpus (**relative frequency**, as in *x words per million* or *x percent*) can thus separate the words that are frequent from those that are rare.

The definition of word can vary from study to study. They can be, for instance, tokens, types, lemmas, or word families.

¹ <https://goo.gl/gzte3J>. Accessed on 08/17/2018.

4.1 WORD FAMILIARITY AND WORD FREQUENCY

Before going further, some clarification is needed. The term *word familiarity* has been used interchangeably, in a somewhat imprecise manner, with the term *word frequency* in research – see, for instance, Aziz et al. (2010), Begeny and Greene (2014), and Leroy and Kauchak (2014). We argue that they are not the same.

Word frequency is simply a type of count. The concept of word familiarity, on the other hand, assumes that the most frequent words are also the most familiar, which may not always be the case (Bailin and Grafstein, 2001, p. 287), given the differences in people’s backgrounds, as discussed in section 2.4. For instance, the word *aardvark* may be familiar to biologists yet completely unfamiliar to non-biologists. We will thus dispense with the muddled term “familiarity” and use *frequency* instead, unless we are citing or discussing a particular study that has used the former, for fidelity to the original text.

4.2 SCIENTIFIC EVIDENCE

In the following sections, we will outline and discuss a few studies that investigate the influence of word frequency in actual difficulty. It will be possible to discuss them individually, for they are the only ones we have found. In fact, the most recent – Leroy and Kauchak (2014) – even claims to be the first of its kind (i.e. to investigate actual difficulty in relation to word frequency).

4.2.1 *Kandula et al. (2010): lexical simplification*

Focusing on electronic medical records and academic articles, Kandula et al. (2010) explored the concept of **lexical simplification** by replacing difficult terms with easier synonyms (making the assumption that more frequent synonyms are easier). Cloze scores showed a statistically significant improvement after simplification.

Despite the improved cloze scores, it is hard to establish a clear link between frequency and difficulty in this particular study as the authors included other types of simplifications to deal with “difficult terms”: explanation generators, syntactic simplifications,

and further simplifications to grammar by using part-of-speech taggers and cohesion estimates (p. 368). In addition, only four subjects were tested with the cloze tests (which have their own methodological problems, as we have discussed).

In the words of the authors (p. 369), “although all the improvements are statistically significant, the magnitude of improvements is rather small”.

4.2.2 *Leroy, Kauchak, and Mouradi (2013): lexical simplification*

This study by Leroy, Kauchak, and Mouradi (2013), conducted over the Internet with 187 subjects, aimed to test four different types of text for actual and perceived difficulty (tested with Likert scales, multiple choice questions and cloze tests): the original unmodified text; lexically simplified text (changing less frequent words to their more frequent synonyms); coherence-enhanced text; and text that was both coherence-enhanced and lexically simplified. According to the authors, coherence is improved by “ensuring that no gaps exist in the flow of a document by use of anaphoric referents, connective ties, synonyms, etc” (p. 719).

Lexical simplification was found to reduce the perceived difficulty of texts, whereas coherence enhancement reduced actual difficulty (measured by multiple choice questions). The cloze tests, on the other hand, showed that “lexical simplification can negatively impact the flow of the test”, and that “for all types of words, the participants performed better with text that was not lexically simplified” (Leroy, Kauchak, and Mouradi, 2013, p. 726).

4.2.3 *Leroy and Kauchak (2014): word frequency influences difficulty*

Possibly the largest study to this day to investigate actual difficulty of words, Leroy and Kauchak (2014) used 239 subjects over the Internet.

As a basis for frequency counts, the authors used a large corpus (the Google Web Corpus², with over 13 million unique 1-grams, and a word list with 64,000 common English dictionary words (reviewed manually to exclude proper names, number-letter

The texts chosen were: eight sentences on smoking cessation, each taken from separate texts, and two abstracts that were not descriptions of experiments. They were found via PubMed search results with the query smoking cessation.

² https://cogcomp.org/page/resource_view/69. Accessed on 08/17/2018.

In addition to actual difficulty, the authors also tested subjects' perceived difficulty through a Likert five-point scale ranging from very easy (1) to very difficult (5) and found a correlation between perceived difficulty and word familiarity: the words that were perceived as most difficult were among the least frequent.

combinations, internet-specific syntax, and formulas). Then, they selected 25 words from each threshold or percentile of frequency: 25 words taken randomly from the top 1 percent most frequent words, then another 25 words from the 9-10 percent most frequent words and so on, until the 99-100 percent most frequent words, for a total of 275. Pairing the correct definition (a rephrased, simplified definition from WordNet) of each of those 275 words, they found that **actual difficulty was correlated with word familiarity** (i.e., word frequency):

the results show that words with a lower frequency of occurrence (higher percentile) are more difficult and less often correctly defined by participants (p. e171),

as shown in fig. 4.1:

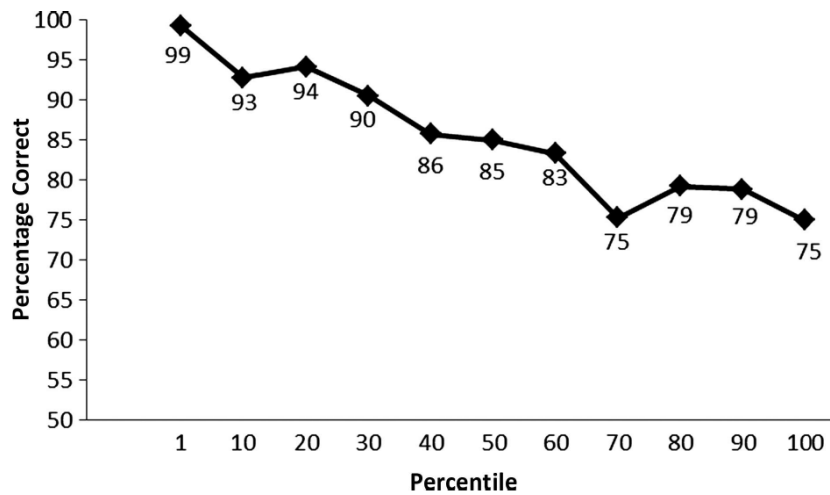


Figure 4.1: Average actual difficulty for words grouped by word frequency of occurrence, from Leroy and Kauchak (2014, e171).

Reinforcing the criticism of Type A (orthography-based) readability formulas mentioned in chapter 3, Leroy and Kauchak (2014) state that no relationship was found “between the word length and actual difficulty” (p. e171).

4.2.4 *Martinez-Gómez and Aizawa (2013): a Bayesian approach*

Constructing a Bayesian causal network based on three different corpora – Simple Wikipedia, the English Wikipedia and PubMed³, Martinez-Gómez and Aizawa (2013) investigated the correlation of 22 different linguistic features to fixation times (measured on 40 participants – most of which were non-native English speakers – with an eye-tracker while they read three texts each), the assumption being that the longer the eye fixates on a specific word, the more cognitive effort (and therefore difficulty) is involved. Their findings are interesting to us:

According to the cognitively-grounded reading difficulty, lexical perplexity (surprise), the occurrence of named entities, out of vocabulary words, passive clauses, academic words, nouns and abstraction (hyponyms) are the linguistic features that required longer fixation times in order to understand those documents (Martinez-Gómez and Aizawa, 2013, p. 1389).

Despite not investigating word frequency directly, nearly every linguistic feature found to correlate with longer fixation times in this study has a connection to frequency: out of vocabulary words (words that were not present in lists of highly frequent words); occurrence of named entities (bound to be less frequent than non-named entities like common nouns); and “academic” words, nouns and abstractions (which are arguably more complex and less likely to be frequent).

4.2.5 *Summary*

A clear-as-day relationship between word frequency and text difficulty cannot be established. There are not enough studies to say conclusively that higher frequency equates to less difficulty and vice-versa. Two of the studies analyzed – Kandula et al. (2010) and Leroy, Kauchak, and Mouradi (2013) – provided either inconclusive or neutral results in regards to difficulty. Especially

Perplexity and surprise are measurements in probabilistic language models. In essence, the greater the deviation from the word that is expected (probabilistically), the greater the surprise and perplexity (Lopopolo et al., 2017).

³ <https://goo.gl/9zWNzK>. Accessed on 10/01/2018.

when considering the understanding of passages of text, lexical simplification – replacing lower frequency words with higher frequency equivalents – does not seem to yield significant results, at least on the health-related texts used in the studies.

On the other hand, the difficulty of individual words does appear to be affected by frequency, according to a large study by Leroy and Kauchak (2014). Still in regards to individual words, the experiment by Martinez-Gómez and Aizawa (2013), with a completely different approach (equating longer fixation times at word level to greater difficulty) seems to reinforce these findings.

When writing a text, context can be construed so as to lessen the difficulty of the individual words it contains, by clarifying lower frequency (rare) words through, say, didactic explanations and restatements that are composed of higher frequency words; in other words, one could say that the argument that *individual* words can be difficult does not allow us to say, in turn, that sentences, paragraphs, and entire texts are difficult.

We do agree with this statement, especially considering that there is a persuasive case to be made that the semantic content of words is mainly underspecified, with context and use filling the semantic gap (Jaszczolt, 2005, p. 4). We can attempt to address this discrepancy between individual word difficulty and text difficulty in two ways.

First, we can obtain median values of difficulty for *all* the words in a given text, i.e. make an appraisal of difficulty of all tokens combined, even if repeated, instead of only looking at the unique (different) words in a given text. We will discuss this approach in more detail in section 12.2, p. 85.

Secondly, in addition to the individual word level, we could look at sequences of words, capturing, for instance, phrasal verbs and other types of formulaic language which may often have differing frequencies (and, we suppose, difficulties) than its individual parts. This we will discuss in chapter 5.

4.3 WORD DISPERSION

The potential correlation between word frequency and word difficulty may, in fact, not be due to frequency, but instead to contextual diversity (CD) (also called *context variability*). Especially in the corpus linguistics literature, CD has been “neglected or even completely ignored” (Gries, n.d., p. 10).

Contextual diversity refers to the different instances where a word can be encountered, i.e. “the number of contexts in which words are experienced” (Adelman et al., 2006, p. 814), also defined as “the environment in which the stimulus [e.g., a word] was encountered” (Shiffrin and Steyvers, 1997, p. 760).

Initially, this concept appears to have been used in first language acquisition research, in terms of the amount of different contexts a child finds themselves in (different conversations at home; conversations at school; at the doctor; while playing with friends, etc.), with the definition of “context” varying in the literature (Shiffrin and Steyvers, 1997, p. 760).

Later, the same term, CD, was applied in corpus linguistics to refer to “the number of passages (documents) in a corpus containing the word” (Brysbaert and New, 2009, p. 984); the same concept has been called **word dispersion** (as defined in Okamoto, 2015, p. 3: “how evenly a given word is spread across a corpus”), which, for reasons of clarity, we will use. However, we will keep the term contextual diversity or CD when discussing the literature in which that term appeared.

4.3.1 *Word frequency vs. word dispersion*

A word that is very frequent may not necessarily be useful for a learner. We will illustrate this with an example.

Suppose a word frequency count was performed for a text collection comprised of 1,000 different texts in English on various subjects (art, architecture, biology, etc.).

With a frequency word list generated, we see on rank 500 the word *manuscript*. Anyone who has ever inspected a frequency word list would not expect this word to appear among the 500 most frequent in the English language, whereas *value* or *building* probably would.

Value and building are placed near rank 500 on a list of the most frequent words in English, according to COCA.

However, while investigating the text collection, we see that two of the 1,000 texts – less than half a percent of all texts—are catalogs of 15th century illuminated *manuscripts*. The words on those texts are repeated constantly because of their subject matter—in a similar way as we, in this work, have repeated the words *word*, *difficulty*, *language*, *learner*, *frequency*, and so on, thus increasing the frequency of those words to an inordinate degree. Despite the high frequency in that particular collection of texts, the word *manuscript* is not very useful, given how circumscribed it is to a particular domain.

Gries and Nick C. Ellis (2015, p. 232) describe the difference between frequency and dispersion neatly:

(...) frequency answers the question “how often does *x* happen?” whereas dispersion asks “in how many contexts will you encounter *x* at all?”

4.3.2 *Scientific evidence*

From the early 2000’s, researchers in the field of cognitive linguistics and psycholinguistics started paying more attention to contextual diversity in regards to its effect on word recognition and the speed of lexical access.

A careful study conducted in 2003 with over 141 participants, controlling for many variables (the degrees of ambiguity and concreteness of words, the strong correlation between CD and word frequency (WF), the clustering properties of the corpora used, etc.), found that

CD predicts word-processing times independently of WF and, moreover, that there is no evidence for a facilitatory effect of WF independent of CD (...) CD had a unique effect, with high CD leading to fast responses, whereas WF had either no unique effect or a suppressor effect, with high WF leading to slow responses (Adelman et al., 2006, pp. 815, 821)

These findings were replicated in a lexical decision experiment for young readers, again showing an effect of word dispersion, but not word frequency, in word identification times (Perea et al., 2013).

Also in first language acquisition, CD seems to be an important factor, with “the earliest learned words [being] the most contextually diverse in the learning environment” (Hills et al., 2010, p. 259). In addition, in terms of selecting words to include in pedagogical materials, a criterion such as word dispersion seems desirable (Okamoto, 2015, p. 7).

An evaluation of word frequency norms by Brysbaert and New (2009) again found word dispersion to be a better measure for psycholinguistic tests (such as prediction of reaction times, lexical decision, etc.) compared to word frequency.

None of these studies has touched upon word dispersion in terms of language learning in adults; as stated by Okamoto (2015, p. 7), this is a neglected lexical property in teaching: “there is no research that addresses this issue [word dispersion] from the perspective of goal setting for vocabulary teaching”.

Our line of reasoning, if we presuppose a link between frequency and difficulty, now becomes: words used throughout a collection of texts will naturally be less difficult, as speakers are expected to find them more often and therefore have a higher likelihood of learning them. If we count these more disperse words, we will find that they are often the most frequent (there is a strong correlation between frequency and dispersion, especially for the top 6,000 most frequent words, as stated for instance in Steyvers and Malmberg (2003, p. 761) and Okamoto (2015, p. 7)).

Due to the lack of empirical investigation on the interplay between dispersion and difficulty, we can only make an indirect correlation between them through the “third wheel” that is frequency; but in terms of language students, the target audience of this project, word dispersion is arguably more useful than word frequency. A language student will not be confined to a single book, film, album, or TV show, nor a single subject in the target language; they will (or, at least, they should) attempt to understand texts from many different contexts, in order to increase their proficiency in the language studied.

Thus, we argue that the more likely a word is to be encountered – i.e., the greater its word dispersion or contextual diversity – the more important it becomes for learning. We will discuss this in more detail in section 11.1, p. 73.

MULTIWORD EXPRESSIONS

One neglected feature in difficulty-related linguistic research is accounting for multiword expressions (MWEs) such as **bat an eye**, *to kick the bucket*, *in the light of*, etc. These can also be called *phrasemes* or *chunks*, although the definition of a multiword expression is more strict:

any word combination for which the syntactic or semantic properties of the whole expression cannot be obtained from its parts (Sag et al., 2002, originally cited in Caseli et al. (2010)).

Studies on the optimal vocabulary size for language proficiency do not usually take MWEs into consideration: “(...) individual words are all that is mentioned in current research on vocabulary thresholds” (Martinez and Schmitt, 2012, p. 268).

This neglect of MWEs may be due to the fact that obtaining them from corpora is an especially troublesome task. There is even an article that compares the extraction of MWEs to a “pain in the neck” for natural language processing (Sag et al., 2002). A panorama of the MWE-related research is given in Omidian et al. (2017, p. 490), where the authors stress the difficulty in defining proper criteria for classifying and extracting MWEs from texts.

Therefore, MWEs may well be considered an elephant in the room in vocabulary research. Their large number – and consequently their importance – is stated as such:

As Jackendoff (1997) notes, the magnitude of MWEs is far greater than what has traditionally been realized within linguistics. He estimates that the number of MWEs in a speaker’s lexicon is of the same order of magnitude as the number of single words. In WordNet 1.7 (Miller and Fellbaum, 2008), 41% of the entries are multi-words. Some specialized domain vocabulary, such as terminology, overwhelmingly consists of MWEs. (Bu et al. (2011), p. 3)

Several attempts to extract MWEs from corpora have been made, some of them being Granger (2014)'s *lexical bundles*, Simpson-Vlach and Nick C Ellis (2010)'s *Academic Formulas List*, and Martinez and Schmitt (2012)'s *Phrasal Expressions List*. No single list, however, will be definitive, and new MWEs are likely to appear as frequently as new individual words.

To our knowledge, no readability formula or text difficulty measurement has taken MWEs into account. In our opinion, if a correlation can be established empirically between the difficulty of individual words and frequency, it is only natural to suppose that there is a correlation between frequency (arguably as well as dispersion) and the difficulty of an MWE.

In the next section, we will discuss the only study that, to our knowledge, investigates MWEs in regards to text difficulty.

5.1 MARTINEZ AND V. A. MURPHY (2011)

Martinez and V. A. Murphy (2011) tested the reading comprehension of 101 adult Brazilian learners of English with two texts. The first text contained only the top 2,000 most frequent words in English, whereas the second included some multiword expressions made up of these 2,000 most frequent words.

The two texts had practically the same length (416 vs 412 words) and used the same set of words, the only difference being that the second had MWEs. Participants were asked to rate their own comprehension; to mark their times for completion of each tests; and also to answer true or false questions that tested their understanding. Thus we have here a study of the influence of MWEs on both actual and perceived difficulty.

In regards to actual difficulty, the authors found that “participants’ scores were significantly lower on Test 2 [with MWEs added] relative to Test 1 (...) with a strong effect size” (Martinez and V. A. Murphy (2011), p. 278). Time spent in the text with MWEs was also greater (p. 280).

As to perceived difficulty, participants rated themselves as understanding more than their results in the true-or-false test showed, leading the authors to conclude that

learners may believe they understand more than they actually do by virtue of their simply understanding the individual words in a text (Martinez and V. A. Murphy (2011), p. 278),

with around half of the participants overestimating their abilities.

5.2 SUMMARY

From the limited evidence we gathered, it appears that multiword expressions could affect text difficulty. For this reason, and also for being in line with our lexicon-oriented approach, multiword expressions will be included as a factor in our text difficulty estimations. However, our way of accounting for multiword expressions will be different from other studies: instead of struggling to conform to the principles that make MWEs what they are, we will develop a more inclusive approach through combinatorial searches – thus, we will be extracting more specifically *n-gram* data, which is simply a sequence of items from a text sample. Section 10.4, p. 66, lays out our methods in more detail.

POLYSEMY

As previously established, trying to optimize language learning by focusing on a set of the most frequent words has been a trend for the past decades (P. Nation and Coady, 1988; Pigada and Schmitt, 2006). However, an often overlooked fact is that the effort in vocabulary acquisition is potentially compounded by **polysemy**: the different senses or meanings a given lexical unit is thought to have.

The following sections are concerned with the definition of polysemy, how it can be measured or quantified, whether it counts in language learning, and whether it would be feasible to include it in our program.

6.1 DEFINING POLYSEMY

The criteria for defining polysemy can vary. Even the attempt to distinguish polysemy from vagueness is “beset with terminological pitfalls” (Geeraerts, 2010, p. 197). Pragmatically, one could say that the number of senses for any word is infinite (Nunberg, 1978), since different meanings can be devised within a context. M. L. Murphy (2003, p. 18) illustrates that concept with the word *tea*:

(...) let us say that in South Africa I grew to like *rooibos* tea and that I visit Nancy in New York who asks *Would you like some tea?* Now, knowing that Nancy has probably never heard of *rooibos*, I assume that when she says tea, *rooibos* is not a member of the set of things that she intends to refer to, so I reply, *No, I don't care for tea.* For the purpose of this exchange, the sense I use for tea does not include *rooibos*, but in another context I may refer to *rooibos* as tea, as in *The only tea I like is rooibos.*

New meanings used or devised on-the-go pragmatically are, of course, impractical to catalog. However, even with word senses being infinite, it is evidently quite feasible to catalog at least the conventionalized or frequent alternative senses, as dictionaries and thesauri show.

6.2 POLYSEMY AND FREQUENCY

A simpler definition of polysemy as the “number of definitions listed in a dictionary entry” has revealed interesting information. Polysemy seems to correlate to frequency, as Zipf has shown in fig. 6.1.

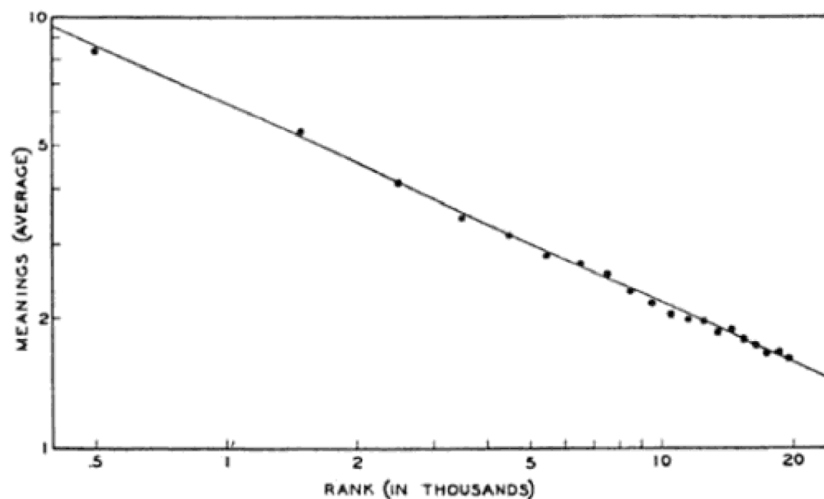


Fig. 2-2. The meaning-frequency distribution of words.

Figure 6.1: Polysemy and frequency. Reproduced from Zipf (1949).

Words used frequently in daily life often seem straightforward to learn or acquire, but in reality they can be packed with different meanings: for instance, the Portuguese verb *quebrar* (break) has at least 13 different definitions on Wiktionary¹. Among them are listed *to rupture*, *fragmentize*, *interrupt*, *transgress* and *become bankrupt*.

¹ <https://goo.gl/Wbjvwm>. Accessed on 11/07/2017.

Thus, polysemy – if Zipf’s correlation between high frequency and high polysemy is to be believed – runs counter to word frequency, since it makes those words that are the most frequent and most familiar to be the most difficult, or at least more time-consuming to acquire. Conversely, the polysemy factor seems to become more unimportant the less frequent a word is.

6.3 COUNTING POLYSEMY

From what we gathered, polysemy could very well be considered as a kind of hidden cost to vocabulary acquisition. Despite the neglect of the subject as applied to language learning, its absence in the field of lexical statistics has not gone completely unnoticed. Word frequency lists have been criticized for disregarding polysemy (Bailin and Grafstein, 2001, p. 288).

There is a practical reason for this, however: accounting for polysemy in word lists needs a significant computational investment. This is probably one of the reasons why automated translation is still imperfect; and considering the many multiword expressions, such as idioms and formulaic language, that can be used, the task becomes even more complicated.

6.4 ACCOUNTING FOR POLYSEMY

Should we attempt to account for polysemy in our project? The ideal, computationally speaking, in accounting for polysemy in texts would be separating the frequencies or dispersions of each sense of a word or expression. However, identifying such senses in text – especially large collections of texts – is at best very time-consuming, as previously discussed.

Alternatively, we could count the number of meanings of each word in a dictionary. The more meanings it has, the more difficult it could be considered, as logically it would take more time for a language student to learn five meanings than one.

For that approach, nevertheless, there is the issue already mentioned of polysemy being predictable (in regards to it being proportional to word frequency, i.e. the greater the frequency, the greater the polysemy). Therefore, not only there would be not

much use in counting meanings in dictionaries, but also our application would become less flexible to other languages, as such a polysemy count (which is not easy to do) would have to be performed by using each language's dictionary.

6.5 SUMMARY

Similarly to multiword expressions, polysemy is a somewhat neglected area of vocabulary and readability research which needs to be addressed and investigated. However, until software can quickly and accurately distinguish word meanings from texts and show statistical measurements for them – something that is simply too complex to do well today – including polysemy counts in difficulty scores makes itself unnecessary due to its predictable nature in relation to frequency. Thus, due to these practical reasons, we will not include polysemy in our proposal.

Part II

METHODOLOGY

The existing approaches for text difficulty analysis, our criteria for our text collection, the rationale behind what will be counted as a word, experiments on n-gram extraction, and statistics on the Wikipedia text data.

EXISTING APPROACHES

Here, we will briefly explore the existing tools aimed at text difficulty analysis. Those that are available online are scarce and little known. We were able to find five of them, summarized in the next sections. Then, we will delineate the principles of our own approach.

7.1 READ-X

Read-X (Miltakaki and Troutt, 2007) evaluates reading difficulty of web text through an assortment of Type A formulas (non-WF based): Lix (Björnsson, 1983), Rix (Anderson, 1983), and Coleman-Liau (Coleman and Liau, 1975). The researchers proposed, as future work, the development and inclusion of a new frequency-based readability formula, adaptable to any language; however, to the best of our knowledge, it has not been made available. In addition, Read-X itself does not seem to be online anymore.

7.2 READABLE.IO

Readable.io¹ offers a free trial and paid plans targeted at writers, professional or otherwise, who want to analyze and, based on the results, make changes to their texts. The website employs Type-A formulas (including Flesch-Kincaid) with no WF component.

7.3 WORDANDPHRASE.INFO

WordandPhrase.info² merits mentioning here despite not being a text difficulty analyzer proper, but instead a web tool that accepts strings of text (of up to 3,000 words) and classifies them into three categories: words that are present in the top 500, in the top 3,000,

¹ <https://readable.io/>. Accessed on 08/11/2018.

² <https://www.wordandphrase.info/>. Accessed on 13/01/2018.

and over 3,000 ranks of word lists by COCA³, thus providing a very cursory impression on the overall frequency of the words in the text. It highlights these words with different colors (pink, blue, green, and yellow) to indicate different frequency ranges.

7.4 ATOS

ATOS for Text, and ATOS for Books⁴ are readability formulas of Type B, i.e. they contain a frequency component. They are targeted at monolingual readers of English in a school setting – thus, its real target is editors of pedagogical materials that need to write or fit their text to a specific school grade. It is a commercial application.

Its differential lies in the conversion of formula values to grade levels; this conversion appears to have been performed with reading performance data from actual students, although there appears to be a lack of independent investigation of ATOS's effectiveness (Benjamin, 2012, p. 68).

7.5 DELITE

Made for German, DeLite (Tim vor der Brück, 2008) uses supervised learning for readability checking that were later validated by ratings of 300 users on a 7-point difficulty Likert scale (thus, its validation is on perceived, rather than actual difficulty). Forty-eight features were included, mainly of a semantic or syntactic nature (for instance, center embedding depth of subclauses). No corpus or text data appears to have been used, and it does not seem to be available online, although it had once a browser-based graphical interface that included highlighting of difficult passages.

³ <http://corpus.byu.edu/coca/>. Accessed on 08/08/2018.

⁴ <https://goo.gl/yG7GFo>. Accessed on 08/08/2018.

7.6 OUR APPROACH

Practically all of the existing tools we have outlined are based on readability formulas which have little to no scientific backing, especially linguistic scientific evidence; have limited or no support for other languages; and are targeted at monolingual readers, usually children. In addition, some seem to no longer be available for download.

Our own approach is meant to be more flexible. Through the Python library Natural Language Toolkit ([NLTK](https://www.nltk.org/))⁵, we offer built-in support for 17 languages.

For languages with non-Latin writing systems, we also offer as an alternative a collection of tokenizers and language models from the ICU (International Components for Unicode) Project⁶, which through the Python library `polyglot`⁷ allows tokenization and sentence segmentation of over 165 languages, among other uses, such as part of speech tagging, language detection and even transliteration. We will demonstrate our proposal on non-Latin writing systems in chapter 14.

In addition, users can train our program on any collection of texts they want, for their own specific purposes; and, considering that many users may have no data of their own, we provide a Wikipedia ([WP](#)) random article extractor for any language, in order to serve as a text collection generator.

7.6.1 *User requirements*

Our original goal was to build a friendly user web interface, to be used with both computers and mobile devices. However, due to time constraints, we focused on the Text Difficulty Scale (which we will soon discuss), and to provide Jupyter Notebooks (more user-friendly versions than raw Python code) in a GitHub page⁸.

⁵ <https://www.nltk.org/>. Accessed on 08/08/2018.

⁶ <http://site.icu-project.org/>. Accessed on 09/13/2018

⁷ <https://polyglot.readthedocs.io/en/latest/index.html>. Accessed on 09/13/2018.

⁸ <https://github.com/filiperubini/text-difficulty-analyzer>. Accessed on 09/15/2018.

As to what type of computer knowledge is expected from users to use our programs, some Python programming experience would be desirable; however, we argue that it is within the ability of most computer users to use our Jupyter Notebooks successfully.

THE TEXT COLLECTION

Before continuing, a brief justification on terminology is needed. For this project, we prefer the term **text collection**, **text data**, or simply **data** over *corpus*, since our efforts in this area are meant for the specific purpose of this work and thus do not aim to be a representative sample of the English language, a complex undertaking in and of itself. Users of our program are encouraged to use whatever texts they prefer, in English or other languages, and use it for their specific purposes. For instance, those interested in improving their conversational abilities in a target language can download subtitles from OpenSubtitles³ of talk shows in the target language.

8.1 SIZE

We have mentioned in section 4.3.1 that word dispersion may be as important a factor as word frequency when analyzing text difficulty. A study by Brysbaert and New (2009) suggested guidelines for building corpora appropriate for word frequency (WF) and word dispersion (WD) counts which we summarize below:

- “A corpus consisting of a large number of small excerpts is better than a corpus consisting of a small number of large excerpts (...) samples of a few hundred words to a few thousand words are better than samples of 100,000 words or more, since many words will appear in all samples” (Brysbaert and New, 2009, p. 987);
- A corpus built for word frequency measurements should have at least 3,000 different texts with “presumably not much gain to be expected above 10,000 samples” (Brysbaert and New, 2009, p. 987);
- “For most practical purposes, a corpus of 16–30 million words suffices for reliable word frequency norms. In particular, there is no evidence that a corpus of 3 billion words is much better than a corpus of 30 million words” (Brysbaert and New, 2009, p. 980);

As discussed, NLTK provides built-in sentence tokenization capabilities for 17 languages (in the case of the module Punkt²) and the ability to train Punkt on any language. You only need a small sample of the language you want to train for (less than one million words is sufficient).

³ <https://goo.gl/J9khTm>. Accessed on 06/19/2018.

These findings regarding corpus building were sought to create better frequency norms for psycholinguistic research (e.g., word retrieval and response tests). For this work, however, we need as much lexical diversity as possible, and although the first three items are helpful for us in regards to word dispersion, the fourth is not – we need to be able to identify most of the expressions in the texts analyzed by the users, and thus we need a text collection larger than 30 million words.

8.2 TEXT SOURCE

We have chosen the English Wikipedia (WP)⁴ as our source of data.

There are disadvantages in using WP as a sole source. Despite being about many different subjects, WP articles could be considered a single textual genre, and the website tries to enforce a style guide⁵ that aims for avoidance of the use of contractions like *wasn't*; gender-neutral language (avoiding the generic *he*, for instance); no preference for any national variety of English, which may split statistical counts for words that mean the same thing but have different orthographies (*honor* vs. *honour*), and present different words used to mean the same thing in different places, like *queue* (UK) vs. *line* (US). On the other hand, the presence of different varieties of English could be considered an advantage in terms of lexical diversity.

One might also think that WP is too formal; editors, however, attempt to make texts “understandable to as many readers as possible” without, at the same time, making them too superficial. Overly technical language use is discouraged, and short sentences are preferred, with the suggestion for editors to use “language similar to what you would use in a conversation” (Wikipedia contributors, 2018a,b).

Our reasons for using WP include:

- It presents an unparalleled range of terminology, jargon, and lexical items, from subjects as varied as pop culture and nuclear physics;

⁴ <https://www.wikipedia.org/>. Accessed on 09/19/2017.

⁵ https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style. Accessed on 06/19/2018.

- It is composed of articles, which are usually short texts with differing subjects – providing a better measurement of word dispersion;
- It is open-source and free to use;
- It is amenable to massive data extraction, with periodical downloadable archives (dumps).

8.3 PROCESSING THE DUMP FILE

A WP article “dump”, a large file which contains all WP articles from its advent until 20 April 2018 was downloaded⁶ and extracted into 124 separate folders using WikiExtractor, a Python library designed for this purpose⁷. Then, files in these 124 subfolders were extracted to separate text files. The result were around 5.6 million different text files.

For the minutiae of the process, including the original code, see the Appendix, sections A.1.1 and A.1.2.

8.4 GOING BIG OR GOING DEEP?

It is possible to work with the *entire* WP dump to build our text data; however, a great deal of computing power (and time) are required.

We have neither; in addition, designing and building the software (our text difficulty analyzer) that is flexible to many languages has always been our priority, rather than building a large and representative corpus to accompany it. Considering that the corpora available online do not usually provide the original text files, making *n*-gram extraction impossible, together with the need of “test data” for the demonstration of the software, we were left with the task of building our own text collection.

We sampled articles from the *entire* WP dump (and not only the first files, which seem to be ordered by creation date, from oldest to newest). This was time-consuming but ensured that articles written from the beginning of WP (January 15, 2001) to April 2018 be included.

Just to give a sense of the range of subjects that can be found in the randomly selected articles, the following are the first twenty-five. The URL address to the complete list of articles in text format is on page 127, section A.1.2.

⁶ <https://dumps.wikimedia.org/enwiki/20180420/>. Accessed on 05/15/2018.

⁷ <https://github.com/attardi/wikiextractor>. Accessed on 05/15/2018.

- | | |
|----------------------------|--|
| 1. Detroit Tigers | 14. Demographics of the Cayman Islands |
| 2. Bookkeeping | 15. Bilinear map |
| 3. Deuterocanonical books | 16. Yangtze |
| 4. Benjamin | 17. Endosymbiont |
| 5. December 3 | 18. Film festival |
| 6. Goddess | 19. Capetian dynasty |
| 7. Great Pyramid of Giza | 20. Electronic mixer |
| 8. BCD | 21. Brazilian Armed Forces |
| 9. Biochemistry | 22. Dayton Ohio |
| 10. Economy of Gabon | 23. George R. R. Martin |
| 11. Covenant-breaker | 24. Context-free language |
| 12. Emperor Seinei | 25. Group homomorphism |
| 13. First Epistle of Peter | |

8.5 TEXT FILE STRUCTURE

Thanks to `wikiExtractor`, the `WP` text files extracted contained only the body of the article texts, save the first line, which includes some metadata info and the title of the article.

As an example, the first few lines of the *Covenant-breaker* text file follow:

```
<doc id="7827" url="https://en.wikipedia.org/wiki?curid=7827"
title="Covenant-breaker"> Covenant-breaker
```

Covenant-breaker is a term used by Bahá'ís to refer to a person who has been excommunicated from the Bahá'í community for the act of covenant-breaking, roughly defined as active opposition to the Bahá'í Faith from a current member. According to Bahá'í law, only the head of the religion, currently the Universal House of Justice, has the authority to declare a person a covenant-breaker.

A person may be declared a covenant-breaker for actions which are seen as challenging the unity of the Bahá'í community, not for personal matters such as failure to obey Bahá'í law or conversion to another religion.

When a person is a declared a covenant-breaker all Bahá'ís are expected to avoid unnecessary association with that person. (...)

One inherent limitation of considering only the text of WP articles is a loss of some non-textual information, such as mathematical formulas (which are usually typeset on the web pages as images), hierarchical elements of lists, and pronunciation guidelines. These may appear as small interruptions in the flow of the text. For instance, in the text we extracted from the article *Capetian dynasty*, the first sentences are rendered as “The Capetian dynasty (), also known as the House of France, is a dynasty of Frankish origin, founded by Hugh Capet.” The empty brackets contained originally a pronunciation guide for the word *Capetian*, (/kə'pi:ʃən/). We argue that this loss of information is incidental, not affecting the cohesion of the articles or its tokenization, as our text cleaning algorithm (to be described further) disallows sequences of non-alphanumeric characters.

CHOOSING A LEXICAL UNIT

Choosing a specific type of lexical unit for frequency and other measurements is a very important issue. If one does not convert the corpus to lowercase, one has, for instance, to count the occurrences of *the* and *The* separately, with the result of inflating the counts. As stated by Reynolds and Wible (2014), a crucial issue in vocabulary-related studies is that a lot of researchers do not enter into detail about their chosen unit of frequency, and the authors urge research in the area to specify (p. 858)

whether tokens were all identical in form, were all inflectional variations of the same lemma, or inflectional and derivational variations of the same word family.

Let us first define some important types of lexical units. In the list that follows, the definition of word family is taken from Bauer and P. Nation (1993) and the remaining definitions are paraphrased and adapted from Marc Brysbaert and Keuleers (2016). Positing the sentence – or, more precisely, the string of characters – “A rose is a rose is a rose” (Stein, 1922) as our example base text, we have:

- **Word family:** “consists of a base word and all its derived and inflected forms” that are morphologically related. We could posit for “A rose is a rose is a rose” the following word families:

“be, being, been, is, am, are, was, were”, etc.;

“a, an”;

“rose, roses, rosy”, etc.

- **Lemma:** Uninflected word from which all inflected words are derived. Also called the “dictionary form” or “citation form”. In “A rose is a rose is a rose”, we have three lemmas: A, ROSE, and BE.
- **Type:** A word form observed in a corpus. In “A rose is a rose is a rose”, there are four types: *A*, *a*, *rose*, *is*. Usually, punctuation is not considered a type.

We will use SMALL CAPS for lemmas.

- **Alphabetical type:** A word form obtained after converting the entire corpus into lowercase and excluding the words formed by sequences of characters other than the letters a-z. In our example, the alphabetical types would be *a*, *rose*, *is*.
- **Token:** A string of one or more characters, usually separated by spaces. In “A rose is a rose is a rose”, there are 8 tokens. Counting the absolute frequency of tokens for each alphabetical type, we have: *A*: 1, *a*: 2, *rose*: 3, and *is*: 2. Sometimes, punctuation and other characters are also considered tokens.

In the next sections, we will examine the advantages and disadvantages of using word families, lemmas, types, alphabetical types, and tokens as lexical units.

9.1 WORD FAMILIES

Selecting word families as a lexical unit requires the assumption that the learner of English as a second language (or any other language, for that matter) will make the necessary semantic connections between the lemma and its morphological variations. Let us examine the word family derived from the lemma DEVELOP, adapted from a table in Bauer and P. Nation (1993), p. 254:

develop, develops, developed, developing, developable, undevelopable, developer(s), undeveloped, development(s), developmental, developmentally, developmentwise, semideveloped, antidevelopment, redevelop, predevelopment

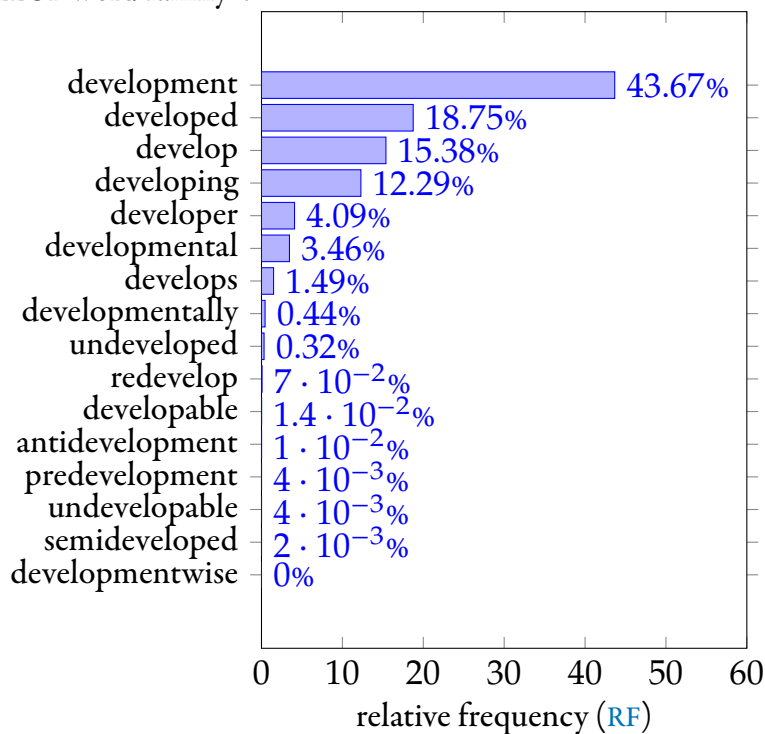
Notice how many forms can be derived from the original word: sixteen. Would knowing the lemma DEVELOP really mean a student has acquired or has the potential to instantly acquire the entire word family? We do not think so. While the lemma DEVELOP roughly means “grow or cause to grow and become more mature, advanced, or elaborate”, the form *developer* refers to, for the most part, a real estate developer, or even a chemical agent used for treating photographic film. The original sense of DEVELOP seems to diminish as we examine the word family further.

Our concerns about using word families are shared by Reynolds and Wible (2014):

For instance, when we read on the paper “the developments since the bombing in June...”, the reference is more to the occurrences after the bombing, not so much the situations that happened because of the bombing.

Brown et al. (2008), for example, has drawn attention to the fact that the use of word family as a unit of counting frequency could be problematic in “that the application of word-building knowledge may have a limited role [for learners] and instead the frequency of individual word forms themselves may determine whether [words] are problematic... or not” (p. 1050). Moreover, Ward and Chuenjundaeng (2009) found L2 learners had difficulties associating related forms of words, especially words with classical or latinete etymologies.

Another important point is that the different word forms in a word family may have dissimilar frequency and dispersion counts. The graph below consists of queries to the COCA regarding the DEVELOP word family¹:



¹ In order to make this plot, we compared the frequency of each word form with the total sum of all word form frequencies in this word family (281,223 occurrences). To simplify the plot, the raw frequencies for the plural forms were combined, so that “development” and “developments” were combined under “development”, and so on. Source: <http://corpus.byu.edu/coca/>. Accessed on 04/24/2017.

As one can see, the frequencies for the first four forms – “development(s)”, “developed”, “develop”, and “developing” – account for over 90 percent of all occurrences. This shows that it is quite unlikely that a learner who knows the word “development” will encounter – and since they can only “acquire” the word by encountering it in some form, also *know* – the meaning and use of the other members of the word family unless they have knowledge of what English suffixes mean.

In addition, the boundaries between base words and derived words is not clear, state Marc Brysbaert and Keuleers (2016, p. 6), adding that “some scholars (...) conclude that the transition from lemmas to word families creates more confusion than clarity (Schmitt, 2010)”.

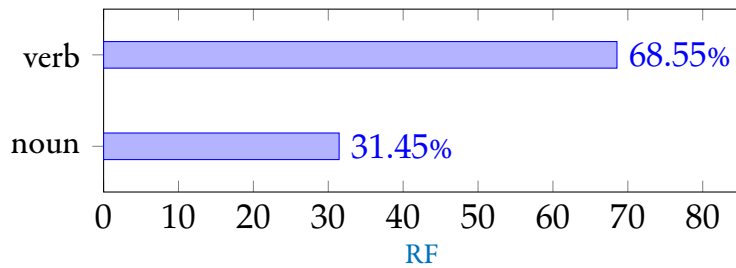
In conclusion, we do not think word families are suitable as a frequency unit for our purposes, as this notion not only presupposes a capability in the reader to connect the word to its “ramifications”, but it also can include word forms that can be distant from its original meaning. In addition, the boundaries between words and their derivations are not a consensus, and we would be limited to predefined lists of word families.

9.2 L E M M A S

Lemmas, at first glance, seem a more reasonable unit of frequency than word families. There would be no boundaries of meaning to consider across words with different parts of speech: for instance, the lemma *develop*, a verb, would have only these possible alphabetical types: *develop*, *develops*, *developed*, *developing*.

One significant disadvantage in using lemmas is that many words in English have identical forms for nouns and verbs – for instance, **play** in “play a game” and in “a theater play” – and those might have differing frequencies. Such a distinction is usually lost in the process of lemmatization. For instance, “play” as a verb seems to have a higher relative frequency than the noun “play”, as shown below².

² Regarding the making of the plot: for the “verb” category, we have added together the frequencies for “play” found in the part-of-speech tags VVI (infinitive), VVo (base form of lexical verb), and VVo_NN1 – which seems to be a composite of tags, classifying the element first as a base form of lexical verb and then as a singular common noun. Similarly, for the “noun” category, we



It is clear that, for the purpose of gauging frequency (or dispersion, for that matter) to a high degree of accuracy, the simplification effected by lemmatization is detrimental.

9.3 ALPHABETICAL TYPES

What if we counted alphabetical types instead? We can readily dismiss this option, since by choosing them as unit of frequency would mean discarding many important words in the English language that contain non-alphabetical characters, as we will see in the following sections. This would, of course, be of little use to our type of lexical analysis, and thus alphabetical types are out of consideration.

9.4 TYPES

Types are the “different word forms observed in a corpus” (Marc Brysbaert and Keuleers, 2016, p. 1). This is a definition that is not only imprecise but also based on both existing inventories of words and inflectional and derivational paradigms.

One would think types to be an imperfect way to measure word inventories if compared to lemmas; yet, a recent study (Brysbaert and New, 2009) found no inherent advantage for lemma frequencies in comparison to word form (type) frequencies in regards to lexical retrieval times (p. 986).

Advantages to considering types would include shorter computational processing times since mainly a simple operation of corpus cleaning and conversion to lowercase would be required.

have added together the tags NN₁ (singular common noun) and NN₁_VV₀, which, analogously to the VV₀_NN₁ tag, appears to be a composite tag defining the element first as a split base form and singular common noun. The number of occurrences of the “verb” category is 99,587, whereas the occurrences for the “noun” category totaled 45,686 occurrences, for a total of 145,273. Source: <http://corpus.byu.edu/coca/>. Accessed on 05/04/2017.

Table 9.1: Occurrences of PLAY word forms in the COCA.

Word	Occurrences	Relative frequency
<i>play</i>	145,360	47%
<i>plays</i>	31,354	10%
<i>playing</i>	68,145	22%
<i>played</i>	64,497	21%

In addition, types are more statistically accurate than lemmas, since the different word forms of a single lemma—in a way similar to what we have seen previously for word families—can have significantly different frequencies and dispersion values. Table 9.1 shows the occurrences for the word forms of the lemma PLAY in the COCA.

Again we see quite an unequal distribution, with the form *play* having nearly half of all occurrences. Accounting for this type of discrepancy statistically would be possible with types, but not with lemmas.

One disadvantage of types, also shared by alphabetical types, would be an inflation of the frequency counts, since types are different variations of a lemma. In our example of the lemma PLAY, the word forms *play*, *plays*, *played*, *playing* are counted separately. This inflated number could not be equated, one-to-one, to the learner’s vocabulary, which is usually estimated in lemmas, although estimating the number of lemmas from the number of types is entirely possible with mathematical formulas.

However, the main disadvantage of using types lies on their definition as “word forms” of a lemma. We would need to determine – according to preexisting lists – what are the “allowed” word forms. Brysbaert and New (2009) have done just that in some of the SUBTLEX word lists: out of 282,000 different strings or tokens found, only 74,000 made the final cut: only the strings that matched that spelling database were included. We argue that this constrains the results enormously and puts a check on linguistic innovation.

With types being so restrictive, we are left with tokens. For our project, we will not include tokens, in their classical definition of simply a sequence of characters without any treatment, since they are by definition unsuitable to our kind of lexical analysis. As stated previously, treating, for instance, *The* and *the* as different words would be incorrect, since the variance in capitalization does not, for the vast majority of cases, differentiate meaning.

Instead, we will discuss our specific definition of tokens, coupled with a discussion of what constitutes a word.

9.5 CHOOSING A LEXICAL UNIT

We will now discuss the different possibilities of lexical unit types and the rationale for our choices.

9.5.1 *What is a word?*

One yet unsolved problem of linguistics is that there is no universal definition of word. Since our medium of analysis is a collection of texts in orthographic form, our definition of word will be closer to a lexicographer's definition, and thus slightly less problematic. Thus, we will not spend time in reviewing all the possible phonological, morphological, and syntactic tests and criteria to define words as laid out, for instance, in Biderman (1999).

There are different ways to try and solve this problem. The first, already mentioned in regards to types, is to adopt lexicographic conventions by considering as words only the lemmas present in the most comprehensive standard dictionaries, such as the Oxford English Dictionary (OED), and then account for all their possible morphological variations.

We have chosen not to proceed on this route; given the ever-changing nature of the lexicon, with new words being created every day, we would rather not be constrained to already existing compilations of words and to the criteria of the compilers (some may favor some words over others for political reasons, for in-

A string is “an ordered sequence of text characters stored consecutively in memory and capable of being processed as a single entity”³

stance). As an example of the lexical richness that would be lost, some abbreviations and words are purposefully misspelled on the Internet, such as *hodl* (for *hold*) in terms of “holding (not selling) bitcoin”³.

Alternatively, we could take an approach similar to alphabetic types: convert our collection of texts to lowercase and consider only the tokens or **strings** containing the minuscule English alphabet. However, there are many words that contain one or more non-alphabetic characters, especially hyphens, as in “mother-in-law” or “self-esteem”.

Thus, let us now consider strings composed by only the minuscule letters of the alphabet and the hyphen. The problem is that in English there are strings composed by digits that are, if not considered as, at least **used** as words:

- “One year after **9/11**...” [*The September 11 attacks*];
- “Andrea wanted to buy a **4x4**...” [*four-wheeled vehicle*];
- “I’m going to take a **101** on programming” [*introductory course*];
- “I need to find a **24/7** ASAP” [*a round-the-clock service*];
- “That was a terrible **3D** movie.” [*three-dimensional, also written as 3-D*];
- “Your eyesight must be **20/20** to qualify.” [*having normal visual acuity*]

The Basic Latin Block in the Unicode, a standard for text encoding, contains all the ASCII characters, i.e. the upper and lowercase 26 Latin letters, numerals, and a number of punctuation marks and symbols.

Additionally, we must also take into consideration that many words in English were borrowed from French and other languages and may contain Latin characters other than those in the Basic Latin Unicode Block, some examples being *née*, *attaché*, *hors d’oeuvre* (take notice of the apostrophe [’], another non-alphabetic character).

Yet another detail to keep in mind is **typographic ligatures**, characters combining multiple letters mainly for stylistic purposes, such as *medieval*, *aesthetic*, or *œuvre*.

³ https://www.reddit.com/r/Bitcoin/comments/2b8t78/whats_hodl/. Accessed on 01/30/2018.

In order to illustrate further the possible complexity of words, character-wise, Wiktionary contains a list⁶, last updated in 2012, of 453 words that begin with non-alphabetic characters, the majority of them digits. Some contain only digits (360, 911), whereas others contain digits and letters (1880s and all “decade” terms), and a number of them contain a combination of digits, letters, and other characters (‘80s and all “decade” word abbreviations). Quite a few are jargon that not many people will encounter in their lifetimes, not even native speakers: 0-6-6-0, 4-6-2+2-6-4, 4-4-2 (different types of locomotives), 5-methoxy-dimethyltryptamine (a hallucinogenic drug), and 8NI (on computing, shorthand for “eight data bits, no parity, one stop bit”); however, the examples we have given previously in this section show that the variety of characters in words people use in their day-to-day activities goes far beyond the scope of the Latin alphabet and most speakers’ pre-suppositions of what a typical word is.

9.5.2 *The learner’s perspective*

9.5.2.1 *Numbers*

In view of this being a language learner-oriented work, we could adopt the learner’s perspective and exclude words they are not supposed to learn. An obvious example is numbers: no learner would be expected to acquire as a word, say, the number 19,892.

However, it is hard to distinguish what makes a number significant or not – is it a year? A quantity? Does it work unequivocally as a “word” (whatever that may be) such as 101 or 007? Numbers can even stand in for words: *2 be or not 2 be, that is the question*.

Our project has a specific complication in that our text data (drawn from Wikipedia) is heavily number and fact-based: years and quantities feature prominently. We have three options: first, include all strings composed by digits; second, include strings that are composed by digits that could function as words on their own, i.e. the “special cases” laid out in Wiktionary; and, finally, barring all strings composed by digits. Pros and cons of each approach are listed below.

⁶ <https://en.wiktionary.org/wiki/Index:English/0>. Accessed on 05/25/2017.

- **Including all digits:** *Pros:* Overrides the problem of relying on already available lists, which are bound to be limited and outdated; thus, could capture special cases that have not been previously catalogued. *Cons:* Our text data can have skewed statistical measurements due to a significant presence of incidental numbers; texts that users send for difficulty analysis which include digits that are not frequent or disperse can have their overall frequency or dispersion lowered artificially, affecting difficulty estimations.
- **Including only special cases:** *Pros:* In theory, would “separate the wheat from the chaff”; the digit strings that are significant versus the ones that are simply incidental, preventing skewing of the measurements of overall text frequency and dispersion. *Cons:* Special case lists are limited and outdated; criteria are necessary to say for certain which digits are significant and which are not.
- **Barring all digits:** *Pros:* Numbers are a relatively small part of texts in general; the amount of special numbers that function as words seems even smaller; our text data WP is heavily number-based, and these numbers are bound to be more incidental than significant. *Cons:* Dismissing a small number of cases that hold significance (*101, 007*, etc.); dismissing a small number of *n*-grams that include digit strings, such as *0800 number* or *1600 Pennsylvania Avenue*.

There are always going to be problems with each approach, but for this particular project’s requirements (in particular the nature of our text data) we will exclude all tokens composed solely by digits, or a combination of digits, commas, or dots. This means that while *1800s* and *1-gram* are included, the tokens *1800*, *19.95*, *3,301*, and *1* are excluded.

However, this restriction to numbers is limited to individual tokens, i.e. 1-grams; for 2-grams and 3-grams, we will not bar numbers, since if we did, we would end up with incorrect *n*-grams, e.g. “at pm” instead of “at 2 pm”. Should one, for whatever reason, need to account for digits in their text data, these exclusion parameters, whether for 1-grams or for longer *n*-grams, can be toggled off.

9.5.2.2 *Proper names*

Proper names could also be a tentative target for exclusion. We argue that proper names of people such as *Julia* or *Andrew* probably do not need to be learned in the same way one learns, say, a common noun, but what about the names of countries, continents, or territories (*the Netherlands, Germany*)? Or the names of languages or country-of-origin (*Dutch, German*), which are also capitalized in English? These must be learned, and there must be differences in frequency among them: a learner would probably encounter the words *British* and *American* more frequently than *Zimbabwean*.

Accounting for and discriminating between these different cases, allowing some and barring others, for a collection of texts that we aim to be quite large, would be impractical. We argue, thus, that it is better to be comprehensive, rather than restrictive as to what words will enter our word lists.

9.5.3 *Tokenization and our definition of word*

In the preceding sections, we have discussed the difficulty involved in defining what would be the best lexical unit for counting – what would count as a “word” – for our project.

We have established that, instead of taking a more restrictive route of relying on previously built lists of lemmas in dictionaries and thesauri, we are going to be more comprehensive in our approach – especially if we consider the cases of words in English composed by digits and other characters, which make it harder for us to simply exclude non-alphabetic strings from consideration. Digit-only strings that denote numerals, dates, or other numeric information will be removed.

Having established that, in regards to the actual process of word separation (i.e., the *tokenization*) process, a requirement we have also made is that **the collection of texts be converted to lowercase first**, in order to avoid duplicates due to sentence or paragraph-start capitalization.

Even though in our algorithm we separate texts by sentences before separating them by tokens, and we could convert to lowercase only the first token of the sentence (which is usually capitalized), it is difficult to ascertain whether this first token needs

capitalization or not; sentences can start with, say, *Well, obviously we have a rapist in Lincoln Park* (where converting to lowercase would be the correct approach) or *Tolkien was a terrific author* (where it would not). We reason that the less we rely on language-specific parameters, the better, in terms of the flexibility of the program for other languages.

Converting to lowercase, of course, introduces an issue with proper names: if they coincide with common nouns, they inflate the counts. For instance, the company name, *Apple*, could affect the counts of the the common noun *apple*. Brysbaert and New (2009) warn of this potential count inflation, and state that “word dispersion is more robust against this type of distortion than is the WF measure” (p. 987).

Aside from numerals, we can exclude without much controversy punctuation and other single non-alphabetic characters. Now, we first need to separate our texts into words, and for this we need a tokenizer.

Instead of building a tokenizer of our own, we will use a freely available tokenizer for English (*Treebank*, available in the NLTK Python library) to break text into sentences and sentences into words. Its documentation⁷ states that, besides considering spaces as delimiters, “most punctuation is split from adjoining words”, “verb contractions and the Anglo-Saxon genitive of nouns are split into their component morphemes” (as in *children’s* being separated into **children | ‘s**).

After the list of tokens is generated, we can proceed to exclude punctuation and some other characters from consideration.

What is left is thus a “word” for the purposes of this project: **tokens that have been converted to lowercase which are not composed of 1) solely punctuation marks or other non-alphanumeric characters; 2) solely digits, or a combination of digits, commas, and dots, indicating numerals, e.g. 95.50, 1,000, or .37.**

⁷ ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html. Accessed on 06/12/2017.

9.6 SUMMARY

In this chapter, we have weighed and compared the different perspectives of which lexical unit to count. There are advantages and disadvantages in every approach, be it in choosing tokens, lemmas, or even word families; thus, we adopted the lexical unit that was closest to our goals and target audience (tokens in lowercase, barring most numerals).

Having this aspect of the work defined, we will perform experiments on n -gram coverage, as well as lay out the overall process of data extraction, from the raw text files to lexical data.

FROM TEXT TO DATA

10.1 EXPERIMENTS ON *N*-GRAM COVERAGE

In order to see whether including up to 6-grams (which seems to be the upper bound for meaningful or conventionalized *n*-grams) was a worthwhile strategy, we made experiments with an initial random sample of 124,000 (around 2 percent of the entire WP dump) and then added an additional 5,000 random articles, for a total of 129,000 (around 2.3 per cent) out of the 5,601,062 text files, of which 117,638 were used, for a total of 63,361,435 individual tokens (after the corpus was converted to lowercase and excluding punctuation and other extraneous characters, but since this was an early stage in the text data processing, we still allowed all digits) and 276,236,083 *n*-grams of 2 to 6 elements.

Table 10.1 shows overall information on our text data from this initial experiment. The criterium for dispersion was simply the number of articles a given token appeared in.

Table 10.1: Overall information on the Wikipedia data in the first run of experiments with 129,000 articles.

	1-grams	2-grams to 6-grams
<i>Articles randomly selected</i>		129,000
<i>Articles actually used</i>		117,638
<i>Unique tokens</i>	1,096,966	215,133,586
<i>Total token occurrences</i>	63,361,435	276,236,083
<i>Hapax legomena</i>	604,513	198,316,021
<i>Ratio of hapax to total occurrences</i>	0.95%	71.79%
<i>Token with the highest frequency</i>	<i>the</i> (5,487,519)	<i>of the</i> (778,163)
<i>Token with the highest dispersion</i>	<i>the</i> (117,638)	<i>in the</i> (86,625)

Using all this data, we analyzed three long (8,000 to 16,000 “words”, as counted by Microsoft Word) WP articles, namely *Citizen Kane*, *Artificial intelligence*, and *Nuclear power*, which had not been a part of our 129,000-article data from the start.

Our observations were that going this deep (up to 6-grams) is quite wasteful: not only there was a huge amount of *hapax legomena* (tokens with a single occurrence) for the n -grams, but also 75 percent of the n -grams extracted from these three WP articles were not found in our text data. On the other hand, only about 2 percent of single tokens (1-grams) in the three articles were not found in our text data (which we deem to be sufficient coverage).

Given that extracting up to 6-grams was inefficient, we decided to find a compromise in coverage: what is the minimum length of n -grams that provides adequate coverage (over 80 percent)?

In order to find this out, we experimented with extracting 2-grams only from the three articles. This yielded a coverage of 56.36 percent – still not enough. Finally, by extracting both 2-grams and 3-grams, we observed 88 percent coverage, which we deemed to be sufficient. Replicating the experiments on a dozen or so other Wikipedia articles that were not originally in our text data, we found the same overall tendency.

This observation – that shorter n -grams are more widespread (and thus, probably more relevant) – reveals that perhaps humans appear to favor shorter combinations of words as compound chunks of meaning, with longer chunks becoming increasingly rare. This is of course just a conjecture, and the English language (and, we suspect, many if not most of the languages of the world) offers us examples of longer chunks:

- 4 elements: *on the other hand, on the basis of, at the same time;*
- 5 elements: *in the aftermath of the, the right thing to do, at the end of the day, the turn of the century;*
- 6 elements: *in the middle of the night; the fact of the matter is, what do you make of this, etc.*

Thus, for practical reasons (including considerably longer times for the computation of frequency, dispersion, and a greater use of computational resources such as memory and storage space), we have found it best, going forward, to limit ourselves to 1-grams (single tokens), 2-grams and 3-grams.

10.2 CONVERTING TO LOWERCASE

The decision to convert the texts to lowercase has been explained in chapter 9. This is accomplished in Python with the `string.lower()` function.

10.3 FROM TEXT TO TOKENS

Fig. 10.1 represents the transformation of an example text into sentences (i.e., sentence tokenization, the division of text into sentences), then into tokens and n -grams, and finally as data to be included on our database, which stores information on frequency and dispersion for all WP articles that were selected. For the Python code, see the Appendix, p. 125.

Sentence tokenization is not a simple matter and can vary from one language to another. We have used the Punkt sentence tokenizer provided by NLTK¹ for our WP articles; for non-Latin alphabets, we will use the tokenizers of the `polyglot` library.

We need to tokenize sentences (and not simply go straight for word tokenization) since the boundaries of sentences are important in extracting n -grams. This is illustrated in fig. 10.2, which shows the differences between n -gram extraction with and without sentence tokenization. Without it, combinations of tokens *across* sentences – in practically all instances unintended by the writer – may occur.

¹ https://www.nltk.org/_modules/nltk/tokenize/punkt.html. Accessed on 06/19/2018.

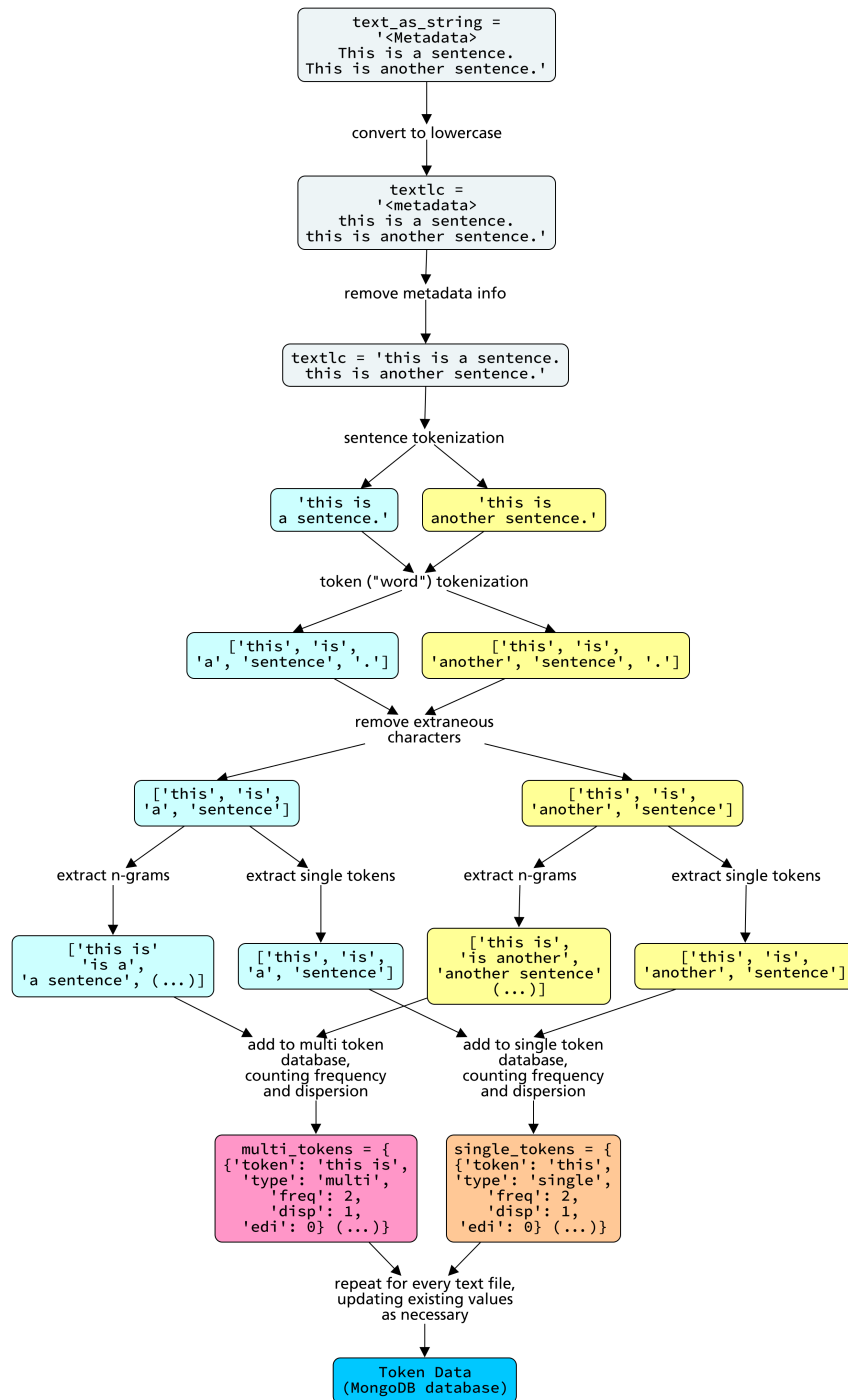


Figure 10.1: Concept map of the process of data extraction.

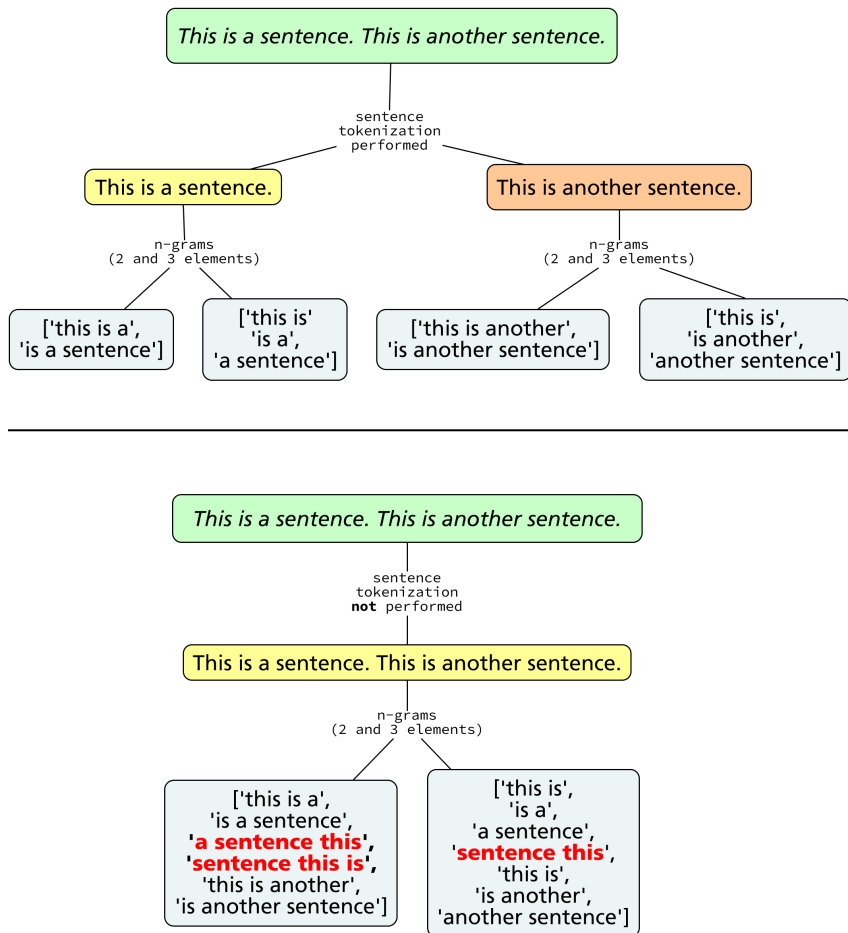


Figure 10.2: Two scenarios of n -gram extraction, with (top) and without (bottom) sentence tokenization. Inappropriate n -grams are shown in red, with strings inside single quotation marks.

10.4 N-GRAM GENERATION

In order to obtain n -grams, we used `intertools`, an iterative library, basing ourselves on the Python documentation recipe `pair-wise`².

Basically, the code will go over each sentence in the text and extract 2 and 3-element combinations in each sentence.

As an example, the code generates the following n -grams for the sentence *jackdaws love my big sphinx of quartz*:

- 2-grams: *jackdaws love, love my, my big, big sphinx, sphinx of, of quartz;*
- 3-grams: *jackdaws love my, love my big, my big sphinx, big sphinx of, sphinx of quartz.*

Evidently, a large amount of what is going to be extracted by this method will not be considered a multiword expression – for instance, frequent phrases like *the man* or *the United States* will be found. We will see further how this approach fares in actual text difficulty analysis.

10.5 THE DATABASE

The database we used for storing our data is `PyMongo`³, which is based on `MongoDB`⁴. We have chosen it for its abundant documentation and its friendliness in terms of storing Python dictionaries as documents, but any other alternative like `MySQL`⁵ would also have done the job.

Using a database and not Python alone frees us from worrying about file management, use of memory resources (which, the greater the number of text files used, the more troublesome it becomes), and allows us to retrieve data quite fast. However, it does require that the user understands the specific syntax of the database, especially for retrieval of data, and that they plan their queries accordingly.

² <https://docs.python.org/3/library/intertools.html>. Accessed on 05/29/2017.

³ <https://api.mongodb.com/python/current/>. Accessed on 08/25/2018.

⁴ <http://www.mongodb.org/>. Accessed on 08/25/2018.

⁵ <https://www.mysql.com/>. Accessed on 08/25/2018.

Our process, in a simplified manner, is to generate one Python dictionary for every token in the text, containing basic information (frequency in the text, the text id, and the length in tokens), and then use the `update_one` PyMongo command to add that dictionary as a document in the database.

After much trial and error with different MongoDB commands, and frustration with what we perceived to be quite a slow process, we found that using `update_one` is the most efficient method, compared to `insert_many`, which is misleadingly fast (since you need to aggregate documents later and this can take much longer), and `find_one_and_update`.

It can process around 34 texts a minute in a 2017 mid-entry gaming notebook, but one can halve that time by 2 by making a copy of the Jupyter Notebook (for another set of texts, of course) and running it at the same time.

The result is one document for every token. Listing 10.1 shows how token data is represented as a document in the database.

Listing 10.1: Example of MongoDB document for the token *serendipitous*

```
{'_id': ObjectId('5b807580c83c11c28d4da39f'),
  'freq': 7,
  'freq_occurred_in': [1, 1, 1, 1, 1, 2],
  'len': 1,
  'occurred_in': ['1164', '3524', '121154', '16972', '20198', '20467'],
  'token': 'serendipitous'}
```

The list `occurred_in` shows the file ids the text occurred in, whereas `freq_occurred_in` shows the frequencies for each corresponding file id. Thus, the token *serendipitous* occurred once in the text files with ids 1164, 3524, 121154, 16972, 20198, and twice for file 20467.

Once you have a document structure like that, you can perform many different queries using MongoDB's commands, as shown in listing 10.2.

Listing 10.2: Different queries for the token database

```
# Get all unique tokens
query = token_stats.distinct('token')
```

```

# Get unique tokens with the length of 1
query = token_stats.distinct('token', {'len' : 1})

# Get unique tokens with length greater than 1
query = token_stats.distinct('token', {'len' : {'$gt':
    1}})

# Get all 2-gram hapax legomena
query = token_stats.find({'freq': 1, 'len': 2})

# Get all tokens that appeared in a particular text
query = token_stats.find({'occurred_in': '1164'})

```

This ends our overall process in extracting data from text and inserting it into a database. In the next sections, we will discuss our rationale behind our difficulty values and how we have implemented them as an algorithm.

10.6 FINAL WIKIPEDIA DATA INFORMATION

Table 10.2 shows overall statistics for the final 124,000 article Wikipedia collection, constructed by the same procedures as described, with the exception of the restriction on numerals and n -grams over 3 elements we already mentioned.

10.7 SUMMARY

In this chapter, we laid out our methodology in regards to sentence and word tokenization, n -gram extraction, and our choice of database. In the next chapters, we will discuss experiments and observations on the interplay of frequency and dispersion, as well as our view of how difficulty at the word and text level should be represented.

Table 10.2: Overall statistics for the final Wikipedia 124,000-article text collection.

	Single (1-gram)	Multi (>1-gram)	
		2-grams	3-grams
<i>Articles randomly selected</i>		124,000	
<i>Articles actually used</i>		123,975	
<i>Median article length</i>	171	171	163
<i>Average article length</i>	497	490	469
<i>Unique tokens</i>	955,472	12,141,174	32,090,069
<i>Total token occurrences</i>	61,560,363	60,642,489	57,934,957
<i>Hapax legomena</i>	520,378	8,714,130	27,287,153
<i>Ratio of hapax to total occurrences</i>	0.85%	14.37%	47.10%
<i>Token with the highest absolute frequency</i>	the (4,681,321)	of the (664,229)	one of the (33,654)
<i>Token with the highest dispersion (DP and number of articles)</i>	the (0.11, 105,796)	of the (0.23, 82,188)	one of the (0.57, 20,184)

Part III

THE TEXT DIFFICULTY SCALE

The interplay between word dispersion and word frequency, the details of the Text Difficulty Scale and Value calculations, experiments on the use of unique versus total token occurrences, and the testing of the scale on texts of different languages.

DIFFICULTY AT THE WORD LEVEL

We have discussed in Part I that the frequency of a word in a sufficiently large corpus appears to affect word difficulty, and that dispersion may be the actual culprit behind difficulty.

Here, we will discuss a few experiments on the interactions between frequency and dispersion, and then make a decision on whether to integrate both frequency and dispersion as measures of difficulty or adopt a single measure.

11.1 FREQUENCY, DISPERSION, AND THE DATA

As mentioned on p. 61, section 10.1, we made some experiments with a 129,000-article version of our text data. In these experiments, we made some queries for tokens in terms of the combination of dispersion and frequency. We then questioned ourselves in terms of what frequency could add, in terms of difficulty, in groups of high and low dispersion words. At the time, we used a simple measurement of dispersion, i.e. simply the number of articles a given token appeared in.

The query for tokens was quite straightforward to perform in terms of tokens that were within a tight range of low dispersion (96 words that appeared in 0.85 percent to 0.93 percent of the texts); however, in terms of higher dispersion tokens, we observed that dispersion values varied quite dramatically from one token to another, and in order to get a sizeable amount of tokens for analysis, it was necessary to query them in a very wide dispersion range (in order to end up with 73 words that were present in 12.1 percent to 67.11 percent of texts), making the analysis of the latter more limited (which is made worse by the strong correlation between frequency and dispersion).

We have delineated four different possibilities, or scenarios, of the combinations of frequency and dispersion; the tokens listed as examples are taken straight from our initial experiments with the Wikipedia text data. To our knowledge, this is the first attempt to investigate all of these four scenarios with an analysis of actual text data.

1. *High frequency, high dispersion expressions*: occur often and in a great number of texts. These mainly include articles, prepositions, conjunctions, and other closed-class words. Examples: *the, of, and, in, to, a, was, is, as, for*.
2. *Low frequency, high dispersion expressions*: occur seldom but are dispersed throughout many texts. We do not think we have found adequate examples for this scenario; as high dispersion is correlated to high frequency, it is difficult to find clear-cut cases of a mismatch between low frequency and high dispersion, and we ended up finding examples of frequency that was slightly lower than the previous scenario accompanied by high dispersion. Examples: *she, her, school, him, city, can, would, these, world, used*.
3. *High frequency, low dispersion expressions*: occur often, but in a limited number of texts. These are the common “false positives” that arise from considering frequency alone as a predictor of difficulty or usefulness. Examples: *squadron, cards, carbon, solar, wine, Singapore, Mississippi, tank, blues, Iraq*.
4. *Low frequency, low dispersion expressions*: occur only a few times in a limited amount of texts. Formal language, scientific terminology and jargon should fit in this category, and out of the four this is possibly the least useful type of expression a beginner can learn, and, we can safely consider, the most difficult of the four. Our analysis of this scenario, however, has yielded examples that were less obscure than we had originally thought: *prevalent, anywhere, consideration, combining, retiring, remarkable, proven, introducing, unfortunately*.

A careful study of more representative corpora is needed to settle the question of whether frequency truly matters. However, in line with Okamoto (2015)'s findings, we contend that frequency does not seem to matter much when accompanying high dispersion; it seems to only be able to distinguish closed-class words (e.g. *the, of, and*) from very frequent, generalized open-class words (e.g. *school, city, world*).

The difference becomes more evident when one looks at low dispersion expressions: lower frequencies seem to produce words that belong to a more formal (and, we may argue based on the evidence, more difficult) register of the language, with tokens such as *prevalent, remarkable, consideration*, whereas higher frequencies produce more likely to be encountered, less formal, and therefore less difficult open-class words (*squadron, carbon, tank*).

Keeping in mind what we have discussed so far, in terms of frequency, dispersion, and difficulty, we can observe two general tendencies in our test data:

- In lexical items of high dispersion, frequency does not seem to affect difficulty to a significant degree.
- In lexical items of low dispersion, it seems that the higher the frequency, the lesser the difficulty.

Based on these observations, our initial idea was to integrate both frequency (which had at least *some* empirical evidence indicating higher difficulty for individual words, as in Leroy and Kauchak, 2014) and dispersion into what we termed an Expression Difficulty Scale, and we had gotten quite far ahead in doing so, by combining a measure of dispersion called deviation of proportions (DP) (Gries, 2008, 2010; Gries and Nick C. Ellis, 2015), which we will outline in a moment, with a measure of word frequency called the Zipf scale (Van Heuven et al., 2014).

However, our attempts to integrate frequency and dispersion were, in a way, positively thwarted by Gries (2008), who has pondered the very same question:

(...) We are now facing a similar issue in dispersion research, namely when researchers and lexicographers also take two dimensions of information – frequency and the effect size of dispersion – and conflate them into one value such as an adjusted frequency...

(...) To say it quite bluntly, this is a mistake because, frequency and dispersion are two different pieces of information, which means conflating them into a single measure loses a lot of information. This is true even though frequency and dispersion are correlated...

(...) Keeping frequency and dispersion separate allows researchers to preserve important information and it is therefore important that we do not give in to the temptation of ‘a single rank-ordering scale’ and simplify beyond necessity/merit – what is needed is more awareness and sophistication of how words are distributed in corpora, not blunting our research tools (Gries, n.d., p. 11)

Further, the author argues that his own proposal for calculation of dispersion, called deviation of proportions (DP), can be more illuminating than just frequency or some amalgam of frequency and dispersion, something that we have touched on before when discussing how dispersion can correct misunderstandings that arise from frequency counts:

In earlier works (Gries, 2008, 2010), the author has cited other actual examples of how misleading frequency without a measure of dispersion can be.

For instance, the products of observed frequency and 1-DP for the two words *pull* and *chairman* in the spoken BNC are very similar – 375 and 368.41 respectively – but they result from very different frequencies and DP-values: 750 and 0.5 for *pull* but 1939 and 0.81 for *chairman*.

Not only is it the dispersion value, not the frequency one, that reflects our intuition (that *pull* is more basic / widely-used than *chairman*) much better, but this also shows that we would probably not want to treat those two cases as ‘the same’ as we would if we simply computed and reported some conflated adjusted frequency (Gries, n.d., p. 13)

Taking Gries' arguments into consideration, we have decided to adopt the criterium of dispersion, in the manner laid out by the author in Gries (2008), as our main guideline for assessing difficulty at the lexical item level, i.e. we are equating the degree of dispersion to the degree of difficulty, where the more dispersed over text data a given n -gram is, the less difficult it will be, and vice-versa.

Since using a single measure seems the more desirable approach, we argue that dispersion is superior to frequency for the reasons discussed beforehand, namely that frequency can be quite misleading (words with same frequency may have wildly different distributions). Another argument for dispersion is that our text data is quite numerous in terms of parts (articles with different subjects), which lends itself better to a dispersion measure. Finally, dispersion is useful especially for our main target audience, i.e. language students, since it appears to predict more effectively how likely a learner is to encounter a given n -gram in a different context, and even seems more reliable for language processing times (Brysbaert and New, 2009).

11.2 MEASURING DISPERSION

As established, we have adopted Gries's measure of dispersion, DP. It was first laid out in Gries (2008), in which the author evaluates all the available measurements of dispersion he could find, concluding that "there are few if any dispersion measures that provide unproblematic measures for equally- and unequally-sized [corpus] parts" (p. 414) and describes DP as having the following advantages: not relying on the unwarranted assumption of equally-sized corpus parts; being neither too nor too little sensitive; ranging from 0 to 1; and can be applied to other kinds of data or scenarios, such as co-occurrence frequencies.

For reasons of clarity, we have adapted Gries's guidelines to the purposes of this work, by changing his notations and terminology. Before outlining the procedure itself, let us define the four different notations needed for the calculation of DP:

- O_{text} : occurrences of tokens or n -grams in each text – for instance, suppose that the Wikipedia article *Housing* has a size of 10,000 tokens, so $O_{Housing} = 10000$;

- O_{data} : total occurrences of tokens or n-grams in the entire text data, i.e. the size of the corpus or text data in tokens or n-grams;
- $O_{token:text}$: occurrences of a given token in a given text – for instance, if the token *the* appears 200 times in the article *Housing*, $O_{the:Housing} = 200$;
- $O_{token:data}$: total occurrences of a given token in the entire data – for instance, if the token *the* appears 1,000 times in the entire data, i.e. across all the articles, $O_{the:data} = 1000$;

The calculation of DP is performed the following way (Gries, 2008, p. 415):

1. Determine $S = \frac{O_{text}}{O_{data}}$ for each text, i.e. the number of occurrences (size) of each text relative to the size of the entire corpus or text data. This corresponds to an *expected percentage*.
2. Determine $F = \frac{O_{token:text}}{O_{token:data}}$ for every token or n-gram, i.e. the number of occurrences of a token in a given text relative to the number of occurrences of this token in the entire data. This corresponds to an *observed percentage*.
3. Compute all pairwise absolute differences of observed and expected percentages, sum them up, and divide the result by two.

The result ranges from 0 to 1, where values close to 0 indicate that a given token or n-gram is distributed across the n WP articles as one would expect, given the sizes of the n WP articles.

By contrast, values close to 1 indicate that a given token or n-gram is distributed across the n Wikipedia articles exactly the opposite way one would expect, given the sizes of the n WP articles.

Let's illustrate this with a practical example. Suppose that our WP data contained 4 articles, each with 250 tokens, for a total of 1,000 tokens.

Now, suppose that the token *pen* occurred 100 times overall in our WP data, with 25 occurrences for each article.

Let us calculate the DP for the token *pen*:

- Sizes of WP articles normalized against overall frequency are $S = \frac{O_{text}}{O_{data}}$, therefore $S = \frac{250}{1000} = 0.25$, i.e. each Wikipedia article contains 25 percent of the total number of occurrences, 1000.

- Frequencies of *pen* in each text in relation to the overall occurrences of *pen* in the entire data are also 0.25 each: $F = \frac{O_{the:text}}{O_{the:data}}$, therefore $F = \frac{25}{100} = 0.25$.
- Summing the pairwise absolute differences of expected and observed frequencies, we have the value of 0, so the token *pen* is perfectly evenly distributed throughout our data.

The greatest advantage of DP is that it works well even if the corpus part sizes (i.e. documents, articles, texts, etc.) are unequal, which, in the vast majority of cases, they are. We will now illustrate these cases with some examples.

Let us suppose that our WP articles have different sizes: instead of four evenly divided 250-token parts, we have four articles of 100, 200, 300, and 400 tokens, respectively, amounting to the same total of 1,000 occurrences. The occurrences of *pen*, however, remain the same: 25 for each article, with a total of 100 occurrences.

Calculating DP for this scenario is shown in table 11.1. In it, the DP value is 0.2, indicating a high degree of dispersion (although, of course, not as high as the previous value of 0).

Table 11.1: Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs equally throughout the corpus.

	S	F	Abs. difference	Sum of abs. dif.	Divide by 2
<i>Article 1</i>	0.10	0.25	0.15		
<i>Article 2</i>	0.20	0.25	0.05	0.40	0.2
<i>Article 3</i>	0.30	0.25	0.05		
<i>Article 4</i>	0.40	0.25	0.15		

Now, in table 11.2, we show an extremely undispersed scenario, with *pen* having all of its occurrences in only one article, with no occurrences whatsoever in the other three.

DP has now increased considerably, to nearly the maximum amount (remember that 1 indicates an extremely unequal, i.e. undispersed, distribution). Notice that Article 1 was the smallest in size, amounting to only ten percent of the corpus.

Table 11.2: Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs in only one part.

	S	F	Abs. difference	Sum of abs. dif.	Divide by 2
<i>Article 1</i>	0.10	1.00	0.90		
<i>Article 2</i>	0.20	0	0.20	1.80	0.9
<i>Article 3</i>	0.30	0	0.30		
<i>Article 4</i>	0.40	0	0.40		

Table 11.3: Example of the calculation of DP in a corpus with unequally-sized parts for a token that occurs only in the largest part.

	S	F	Abs. difference	Sum of abs. dif.	Divide by 2
<i>Article 1</i>	0.10	0	0.10		
<i>Article 2</i>	0.20	0	0.20	1.20	0.6
<i>Article 3</i>	0.30	0	0.30		
<i>Article 4</i>	0.40	1.00	0.60		

In order to show how part sizes affect DP, let us see what happens when we have the same scenario as last time, but now with all the occurrences of *pen* in the largest article in the corpus (Article 4). This is shown in table 11.3. Since in this example *pen* occurs in the largest part, of which it is expected that the largest number of occurrences of *pen* is to appear, the DP value is mitigated in comparison to the previous example.

11.3 CALCULATION OF DP

Our calculation of DP will be performed in an identical manner to Gries's guidelines, with the caveat that overall data sizes will be different for 1-grams, 2-grams and 3-grams (there are different amounts of unique tokens and total occurrences for each n-gram length).

Thus, our calculation of DP for an expression may be laid out as follows:

Table 11.4: The number of unique tokens in each range of DP values in the WP data.

DP	1-grams	2-grams	3-grams
≤ 0.2	6	0	0
0.40	24	10	0
0.60	78	27	2
0.80	359	168	15
0.90	932	572	79
0.92	416	374	52
0.94	698	566	129
0.96	1198	1405	354
0.98	2819	4855	1640
≥ 0.99	948,942	12,133,197	32,087,798

1. An expression is defined as an n -gram of 1, 2, or 3 tokens (lengths 1, 2 or 3). Thus, an expression of *length* = x can only be computed against a text and total data size computed in the same unit (in this example, expressions of length x).
2. Determine $S = \frac{O_{text}}{O_{data}}$ for each text, i.e. the number of occurrences (size) of each text relative to the size of the entire corpus or text data.
3. Determine $F = \frac{O_{token:text}}{O_{token:data}}$ for every token or n -gram, i.e. the number of occurrences of a token in a given text relative to the number of occurrences of this token in the entire data.
4. Compute all pairwise absolute differences of observed and expected percentages, sum them up, and divide the result by two.

11.4 THE DISTRIBUTION OF DISPERSION IN THE WP DATA

The calculation of DP for all unique tokens in the WP data yielded the stats in table 11.4.

The first thing we can note is that the vast majority of tokens is extremely undispersed, with 99.31 percent of 1-grams having a DP value of 0.99 or higher. This is probably due to WP articles presenting quite specific knowledge, with much of that knowledge being constrained to one or only a few articles (one would not expect to find the word *immunoreactivity* in an article on 15th century art, for instance).

Another cause for that disparity is that we are working with tokens, not lemmas, and this spreads tokens out even more: plural nouns are separate from singular nouns, and verb forms are also separate.

Having our criteria for word-level or expression difficulty decided, in the next chapter we will define our criteria for the Text Difficulty Value.

DIFFICULTY AT THE THE TEXT LEVEL

What should a Text Difficulty Scale (**TDS**) composed by Text Difficulty Values (**TDVs**) be made of? Our two main guidelines are that the **TDVs** be applicable to any, or at least most texts without adaptation (preventing multiple algorithms, each for a type of text); and that the algorithm be relatively simple to implement.

With the **TDV** algorithm complete, the best way – as we stated in section 2.3, p. 6, the “gold standard” – to validate its effectiveness would arguably be a large-scale study, subjecting language students at differing proficiency levels to multiple choice questions on the texts they read; then, a multivariate analysis would have to be applied to account for many outcomes. Then, in order to see whether the difficulty value applies to other languages, accompanying studies would have to be performed for these other languages, to see if the principles still hold true.

Unfortunately, the extent of this project disallows such validation to be performed. In the next sections, we will try to reason as much as possible within the available evidence and the theoretical background we have discussed on how best to deal with the problem of text difficulty.

12.1 UNIQUE-TO-TOTAL TOKEN RATIO

Is the number of unique tokens versus token occurrences in a text significant? We investigated the percentages of unique 1-grams (i.e. the different 1-grams that appeared in the text) in relation to total 1-gram occurrences in the 101 most visited English Wikipedia articles¹. They range from 16 percent to 72 percent, with a mean of 26.96 and a median of 26.16 percent.

¹ https://en.wikipedia.org/wiki/Wikipedia:Multiyear_ranking_of_most_viewed_pages. Accessed on 08/29/2018.

The three outliers with higher unique-to-total ratios were all list-type articles, which lay out items without much accompanying text: *List of Presidents of the United States*² (43 percent), *List of The Big Bang Theory episodes*³ (57 percent) and *AMGTV*⁴ (72 percent).

Considering these results, it may be expected of most Wikipedia articles to have a unique-to-total ratio of about 0.25. However, a few non-list articles, like *Pornography*⁵ and *Will Smith*⁶, had higher ratios (around 0.35). The culprit may be that these articles have a greater number of proper names (people, TV shows, albums, places, etc.), rather than a considerably more diverse vocabulary. This, we surmise, is probably an overall tendency of Wikipedia, which tries to be as fact-based as possible, by citing proper names more often than would be expected in, say, an informal conversation, and definitely more than song lyrics – which have a great deal of repetition.

Given that this ratio varies from one type of text to another, we do not think it is plausible to evaluate, say, song lyrics and give them by default a lower difficulty rating than a WP article on Angelina Jolie. It could be that despite the lower than average unique-to-total ratio, one song's lyrics may have very infrequent or undispersed tokens in it; and, conversely, that a Wikipedia article on Big Bang Theory episodes may present a great deal of well dispersed tokens. In keeping with our goal of having the difficulty value apply to most types of texts equally, the unique-to-token ratio would not be an appropriate factor to include, despite the technical ease in implementing it.

² https://en.wikipedia.org/wiki/List_of_Presidents_of_the_United_States. Accessed on 08/29/2018.

³ https://en.wikipedia.org/wiki/List_of_The_Big_Bang_Theory_episodes. Accessed on 08/29/2018.

⁴ <https://en.wikipedia.org/wiki/AMGTV>. Accessed on 08/29/2018.

⁵ <https://en.wikipedia.org/wiki/Pornography>. Accessed on 08/29/2018.

⁶ https://en.wikipedia.org/wiki/Will_Smith. Accessed on 08/29/2018.

12.2 OCCURRENCES AND UNIQUE TOKENS

We mentioned in section 4.2.5 that context should not be neglected when analyzing texts, i.e. one should not look exclusively at the list of unique tokens, since a higher number of occurrences of more frequent or dispersed tokens in relation to undispersed tokens could have an effect in text difficulty estimation. Let us explore this in more detail.

We analyzed a quite long (22,000-word) WP article entitled *Miscegenation*⁷ for the 1-grams appearing in the article and their corresponding DP values.

Fig. 12.1 shows histograms for the unique n -grams and total 1-grams occurrences and the ranges of DP values they belong to. For clarification, these are some examples of tokens found in the *Miscegenation* WP article from each range of DP.

- 0.11—0.26: *the, and, of, from, an;*
- 0.26—0.41: *other, this, after, also;*
- 0.41—0.55: *including, later, before, since;*
- 0.55—0.70: *people, along, work, named;*
- 0.70—0.85: *status, university, particular, evidence;*
- 0.85—1.00: *considerable, historian, substantial, miscegenation.*

If one considers only the unique 1-grams, one can observe how composed by undispersed tokens the article is, with only 2.26 percent of tokens belonging to the bottom half of DP (0.11—0.55) and a huge 97.74 percent of tokens falling in the top half (0.55—1.00). In particular, the most undispersed category (0.85—1.00) contains nearly 83 percent of all unique tokens. In this analysis, the median and mean of DP values for individual tokens hover around 0.94, which is quite high.

The 1-gram occurrence histogram shows the same DP value ranges, but now we are looking at the total occurrences instead of the unique tokens. It is a strikingly different scenario, with nearly half (42.4 percent) of occurrences falling in the lower ranges (most dispersed) of DP and a little more than half (57.6 percent) in the

⁷ <https://en.wikipedia.org/wiki/Miscegenation>. Accessed on 08/29/2018.

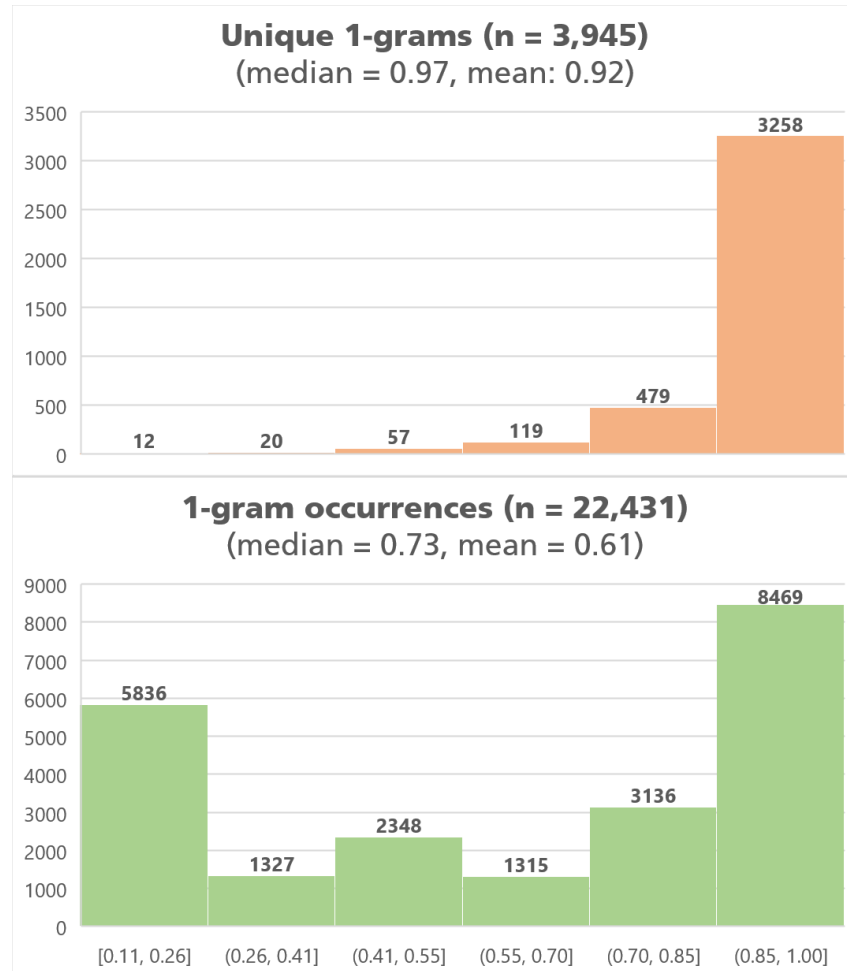


Figure 12.1: Histograms of the number of unique 1-grams and total occurrences of 1-grams (y axis) and their ranges of DP values (x axis) for the WP article *Miscegenation*.

higher ranges. Consequently, the median and mean are lower, at about 0.67. Still, the most undispersed range (0.85—1.00) contains the highest amount of occurrences, about 38 percent; yet, it is a far cry from the 83 percent of the previous histogram.

What can this tell us about text difficulty?

First, that tokens with lower (more dispersed) DP values seem to be less specific or more generalized in content than the less dispersed tokens. For a student, this means that while the strategy of learning the most frequent or disperse tokens first is interesting, considering that they provide greater coverage, they do not seem to provide a strong foundation for understanding the text. In a way, the most disperse, generalized tokens could be compared to a mollusk's shell, whereas the least disperse but more content-rich tokens are the precious pearl inside. This seems to indicate, unfortunately contrary to our assumptions, that there are no true shortcuts for vocabulary learning, as without the pearl there is not much one can do with the shell.

Second, if we were to believe solely in the first statistic of the unique tokens, we would think that the majority of the text was composed by quite undispersed words with DP values over 0.95, like *traditions, referring, convinced, occupation, departure, implemented, definition, coastal, employment, legislation, ruling, favour, overseas*.

Conversely, if we were to believe the alternative offered by the total occurrences, the majority of the text is composed by words with a DP value of around 0.67, e.g. *American, considered, national, include, order, different, according, further, often, small*.

Which of these two options better represents what a reader will need to face? Perhaps a better way to represent this mismatch would be to make an estimate: based on the total occurrences, out of every 10 words, a reader would roughly find:

- Three words like *by, which, also, but, first*;
- Three words like *during, however, years, world, people*;
- One word like *government, development, British, president, therefore*;
- Three words like *typically, pressure, advantage, controversial, imprisonment, ancestry*.

Conversely, based on the unique tokens, out of every 10 words, a reader would find:

- One word like *that, several, period, life*;
- One word like *earlier, today, written, finally*;
- Eight words like *providing, foundation, opposition, essentially, ethnoracial*.

Perhaps the scenario of total occurrences does represent the composition of the text and the experience of the reader more accurately, in terms of quantities, but there is no going around the “pearls” represented by the unique tokens; even if the least dispersed tokens appear less frequently in terms of total occurrences than the unique tokens suggest, they still make up a large portion of the text. And a text with comparatively higher amounts of unique dispersed tokens in relation to unique undispersed tokens will likely present less of a challenge than the opposite scenario.

We argue that it would be best to consider both dimensions as complementary, in the form of the average of the median of unique tokens’ DP values and the median of the total occurrences’ DP values.

12.2.1 2-grams and 3-grams

The difference in distribution between unique and total occurrences of 2-grams and 3-grams is not nearly as striking as the one we observed in 1-grams. The medians and means show this: they have considerably closer values than those of 1-grams, as shown in Figs. 12.2 and 12.3.

These are a few examples of 2-grams in each range of DP values:

- 0.22—0.35: *of the, in the, to the, and the*;
- 0.35—0.48: *with the, as the, the first, in a*;
- 0.48—0.61: *as well, part of, after the, due to*;
- 0.61—0.74: *according to, under the, in addition, along with*;
- 0.74—0.87: *within the, against the, majority of, involved in*;

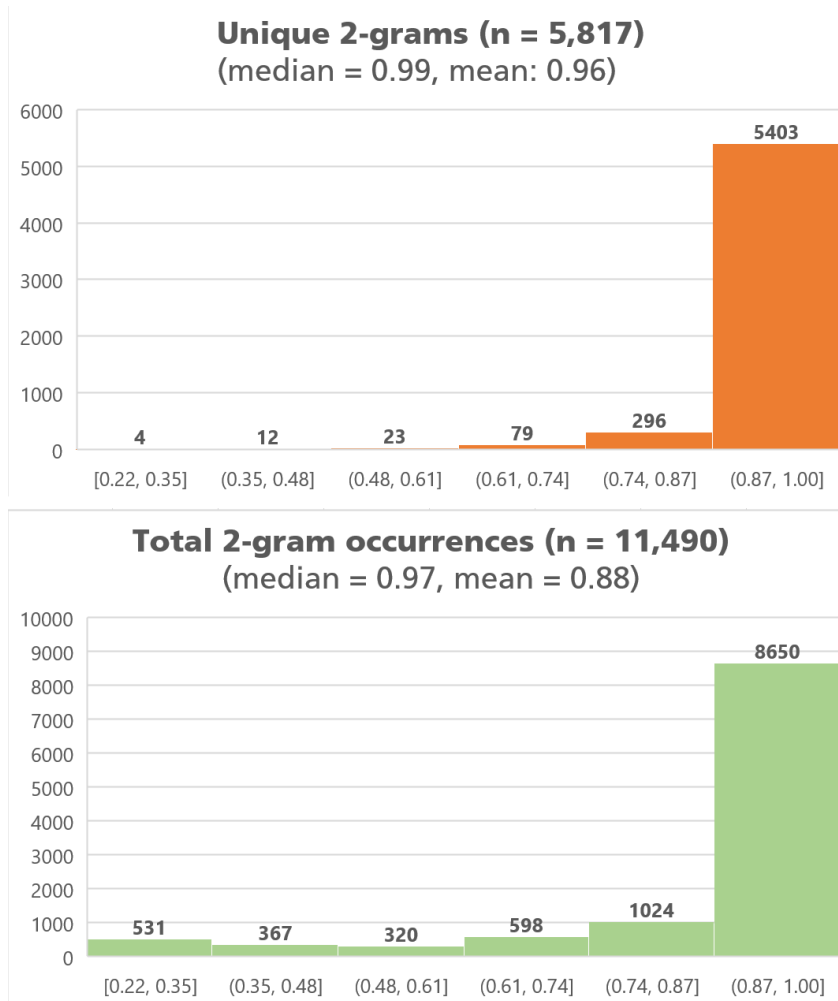


Figure 12.2: Histograms of the number of unique 2-grams and total occurrences of 2-grams (y axis) and their ranges of DP values (x axis) for the WP article *Miscegenation*.

- 0.87—1.00: *in fact, intended to, together with, significant contribution.*

Now, a few instances of 3-grams in each range of DP values:

- 0.57—0.64: *one of the, as well as;*
- 0.64—0.71: *part of the, the end of, the United States;*
- 0.71—0.78: *at the time, in order to, known as the, end of the;*
- 0.78—0.86: *as a result, the same time, the beginning of, because of the;*

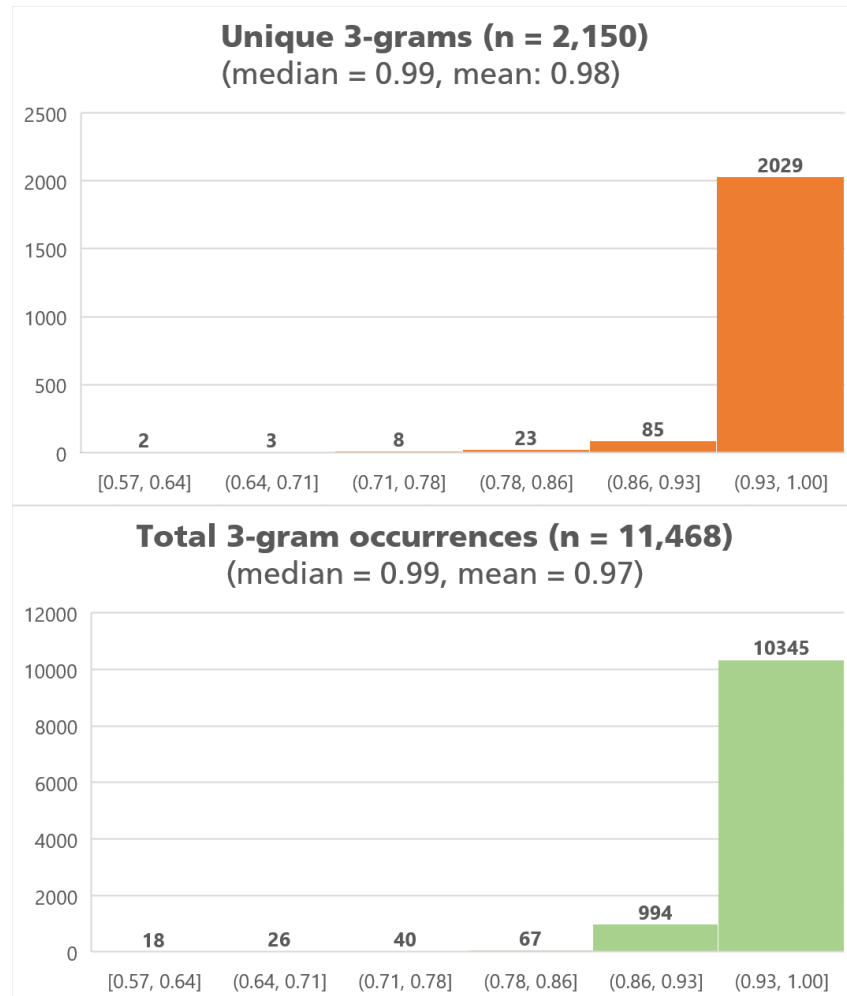


Figure 12.3: Histograms of the number of unique 3-grams and total occurrences of 3-grams (y axis) and their ranges of DP values (x axis) for the WP article *Miscegenation*.

- 0.86—0.93: *the development of, the majority of, an attempt to, associated with the;*
- 0.93—1.00: *New York Times, the Soviet Union, example of how, the Tang Dynasty.*

This is very likely due to the majority of n -grams extracted being text-specific combinations rather than language-wide usage, to the point where we question if even including them is as useful as we originally thought. Notice how the smallest possible DP values increase quite considerably from 0.11 (1-grams) to 0.22 (2-grams) and finally to 0.57 (3-grams), showing again that the greater the length of the n -gram, the less dispersed it will likely be.

Table 12.1: Analysis of 475 WP articles according to their mean and median article lengths, average unique-to-total ratio, and DP.

	1-gram		2-gram		3-gram	
	Unique	Total	Unique	Total	Unique	Total
Median article length	106	188	82	95	26	28
Mean article length	199	512	189	259	63	74
Unique-to-total ratio	60%		77%		87%	
Median DP values	0.88	0.78	0.98	0.97	0.99	0.99
Mean DP values	0.87	0.77	0.98	0.97	0.99	0.99

Since reliable multiword expression extraction is still a problem, a top-down approach – of obtaining compilations of meaningful multiword expressions and looking for them on texts, instead of generating them – seems the best approach when it comes to n -grams of length over 1, even if such lists are not easily available for most languages and would impact the flexibility of our proposal.

While working with the 129,000-article WP data, we observed through the analysis of a few WP articles that these median and mean DP values for 2-grams and 3-grams in texts are consistently higher than those of 1-grams. Therefore, we do not think that placing 2-grams and 3-grams as complementary to 1-grams would do much good, and, in fact, would have the undesirable consequence of bringing TDVs up artificially. There might be some variation from text to text in terms of 2-grams and 3-grams, but they are likely to not be very significant.

In order to be absolutely sure, we checked, alongside *Misc-generation*, a group of 475 articles extracted randomly that are not present in our 124,000-article WP data. The results are shown in table 12.1.

The table confirms our predictions, namely that the 2-gram and 3-gram data provide slight, if any, variation between the number of unique tokens versus total token occurrences, compared to the 1-gram data. Unique-to-token ratios for 2-grams and 3-grams are also considerably higher, as we expected.

In addition, the difference between the median and mean article length for total 1-gram occurrences (188 and 512 tokens, respectively) confirms how variable the unique-to-total ratio is in these articles, and that it should not be relied upon as a factor for gauging difficulty, as we stressed in section 12.1.

12.3 OUR PROPOSAL FOR THE TDV

Taking into consideration our analysis of the data, and keeping with our initial goals, we formulate the Text Difficulty Value as $TDV = \frac{M_U + M_T}{2}$, i.e. the average of: 1) the median of the DP of unique 1-grams (M_U); and 2) the median of the DP of total 1-gram occurrences (M_T), with the median being calculated as $\frac{(n+1)}{2}$, where n is the number of values in an ordered set, with the result representing the x^{th} value. Being a more robust statistic than the arithmetic mean, the median can output the middle point of the data without being overly affected by outliers.

Our Text Difficulty Scale (TDS) thus goes from TDV values from 0 to 1, where 0 is the easiest and 1 is the most difficult.

As a summary, our measure of text difficulty takes into consideration the following factors:

- The median of the different (unique) 1-grams present in a given text, and their corresponding dispersion values, represented by the deviation of proportions (DP), which is to be calculated optimally with text data with many parts;
- The median of the set of occurrences of the unique 1-grams and their corresponding DP values, so that the more frequent a given 1-gram is, the more “weight” will be given to it in the calculation of the median.
- The average of the two medians, which are thought to be complementary in the estimation of difficulty.

12.4 OTHER FACTORS IN TEXT DIFFICULTY

There are, of course, many other factors we could look into. One of them that is relatively easy to calculate would be sentence length, a staple of readability formulas. However, as discussed in chapter 3, these formulas do not present sufficient evidence of usefulness.

There have been a few studies on text difficulty (Hancke et al., 2012; Martínez-Gómez and Aizawa, 2013; Tim vor der Brück, 2008) looking at dozens of different linguistic factors at the same time by using statistical methods, machine learning and natural language processing. These studies are brilliant and rigorous from the perspective of statistics; yet, in our estimation, they do not hold the same standards for their linguistic aspects.

For instance, one of these studies uses a very specific type of text data (Tim vor der Brück, 2008, p. 432):

Our text corpus originated from the municipal domain and differs significantly from newspaper corpora, which are widely used in computational linguistics. It contains a lot of ordinances with legal terms and abbreviations, e.g., *§ 65 Abs. 1 Satz 1 Nr. 2 i.V.m. § 64 Abs. 1 Satz 2 LWG NRW (section 65.1.1 (2) in connection with section 64.1.2 LWG NRW)*. This corpus has been chosen because local administrations in Germany have committed themselves to make their web sites accessible; one central aspect of accessibility is simple language.

The source of the texts is unclear: municipal legislation? Statistical data? A description of the municipal legislature? Educational material for the citizens? How many and how long the texts are?

Not to mention that texts containing sequences like *§ 65 Abs. 1 Satz 1 Nr. 2 i.V.m. § 64 Abs. 1 Satz 2 LWG NRW (section 65.1.1 (2) in connection with section 64.1.2 LWG NRW)* do not seem like good data to base readability or text difficulty estimations on, regardless of whether one thinks that government bureaucrats have achieved their goal in simplifying texts. In addition, the statistical analysis is validated against perceived, rather than actual difficulty, which brings up the possibility of bias, as discussed in section 5.1.

Another study (Hancke et al., 2012) used a myriad of different linguistic factors (many of which were hazily defined, as on p. 5, “noun variation” and “squared verb variation”), assuming that texts of a German magazine aimed at children are less difficult than a corresponding version of the same magazine with texts

aimed at adults. In addition, the authors' conclusions were based entirely on that assumption – that these two types of texts represent two extremes of difficulty – without any sort of validation through either perceived or actual difficulty.

In regards to this adult—children opposition, we argue that these two types of text are potentially more alike than they are different – take, for instance, a traditional text for children like Grimm's Tales, and study the dispersion of the words within against a large, representative corpus. One would probably be surprised at the complexity in comparison to even a newspaper article aimed at adults.

Something similar occurred in Martinez-Gómez and Aizawa (2013). From easiest to most difficult, the sources chosen were the Simple English Wikipedia⁸ vs. the main Wikipedia⁹ vs. PubMed¹⁰, which contains scientific articles on medicine. This division of sources is probably better than the child—adult opposition, although the supposedly less complex lexical profile of the Simple English version of Wikipedia in comparison to the main Wikipedia has been disputed (Silva, 2018).

The lack of validation of both Hancke et al. (2012) and Martinez-Gómez and Aizawa (2013) is a deficiency we will repeat in this project, but we will use a lexicon-oriented perspective that is more language learner-oriented, more backed by the research on lexical richness, and consequently, we argue, less arbitrary.

In closing, one needs mentioning that the more linguistic and statistical factors and features one includes for analysis, the more complex the system becomes to use and implement; the less one potentially knows about the relative importance of each feature (in particular with deep learning models), and the greater the necessity for language-specific models.

8 https://simple.wikipedia.org/wiki/Main_Page. Accessed on 08/30/2018.

9 https://en.wikipedia.org/wiki/Main_Page. Accessed on 08/30/2018.

10 <https://www.ncbi.nlm.nih.gov/pubmed/>. Accessed on 08/30/2018.

12.5 SUMMARY

Having examined the empirical evidence on text difficulty, and described our proposal for assessing text difficulty at the text level, by means of the Text Difficulty Scale (TDS), now we turn to our attempts at testing our proposal with real data, seeing how well it correlates to our expectations.

EXPLORING TEXT DIFFICULTY VALUES

13.1 SPOKEN — ACADEMIC ENGLISH

We are now faced with delimiting clearly different groups of texts according to their assumed difficulty as a baseline, as discussed in section 12.4.

Doing so, however, is more a demonstration than a real test. After all, we will be assuming what is difficult, and our demonstration of the difficulty values will be judged against these assumptions. In other words, the result will not be if our difficulty values are right or wrong, but if they fit with our assumption of text difficulty.

Instead of a child—adult opposition or a simplified encyclopedia article—non-simplified encyclopedia article—academic texts opposition, we will use a spoken—academic text opposition.

Our rationale for selecting this opposition is rooted in lexical diversity: spoken language is less diverse than written language, and this was noticed as early as 1988: “conversations between college graduates more closely resemble a preschool child’s speech to its parents than texts from newspapers” (Hayes, 1988).

From the perspective of the language learner, this means that the minimum size of a vocabulary inventory for understanding and producing speech is significantly smaller in comparison to the vocabulary inventory needed for understanding and producing written texts, a tendency shown in the literature: I. Nation, for instance, cites a minimum of 8,000—9,000 word families for written texts and 6,000—7,000 word families for spoken text (I. Nation, 2006).

Our assumption, therefore, is that the more diverse in terms of the vocabulary a given textual genre or context is, the more difficult it is potentially going to be.

13.1.1 *Spoken data*

For our spoken text data, we have used the Santa Barbara Corpus of Spoken American English (SBCSAE)¹ (Du Bois et al., 2000), which contains 60 different recordings for a total of 249,000 words of “naturally occurring spoken interaction from all over the United States”, with the main form of language use being “face-to-face conversation”. The corpus also documents

many other ways that people use language in their everyday lives: telephone conversations, card games, food preparation, on-the-job talk, classroom lectures, sermons, story-telling, town hall meetings, tour-guide spiels, and more.

For the code used to clean the Santa Barbara, see the Appendix, p. 173, listing C.3, and for a before-and-after example of cleaning, see p. 174, table C.1.

With Python regular expressions, we converted the transcription files of the Santa Barbara corpus to clean text-only files as much as possible. Considering that the transcription conventions are not consistently followed throughout, a few errors may have crept in.

13.1.2 *Academic data*

In order to match Santa Barbara’s size, we built a 250,000-word corpus in SketchEngine² composed by randomly selected academic journal articles from a corpus recently described in Kwary (2018):

The corpus comprises 5,686,428 words, classified into four subject areas: Health Sciences, Life sciences, Physical Sciences, and Social Sciences, following the classifications of Scopus. The words were compiled from 895 journal articles published by Elsevier in 2011–2015.

In each of the four subject areas, there is a multitude of subjects, ranging from medicine and business management to neuroscience and astronomy.

The original text files were available on the corpus website³.

We selected 10 articles randomly from each subject area, for a total of 40 articles. The procedure is shown in code form in section C.2.1.1.

¹ <http://www.linguistics.ucsb.edu/research/santa-barbara-corpus>. Accessed on 08/31/2018.

² <https://www.sketchengine.eu/>. Accessed on 08/31/2018.

³ <http://corpus.kwary.net/database>. Accessed on 08/31/2018.

13.1.3 *Calculating Text Difficulty Values*

With both our spoken and academic data ready in .txt format, we uploaded them as separate “corpora” on SketchEngine⁴ and generated simple frequency lists from them, with the unit of counting being words in lowercase, and proceeded to save them in .csv format.

With a short Python script, we calculated unique and median DP values for each of the 1-grams in the SketchEngine wordlist, and then the TDVs.

See the Appendix, p. 171, listing C.1 for the code used for calculating DP and TDVs for the SketchEngine wordlists.

13.1.4 *Analysis results*

The data related to the spoken–academic comparison is shown in table 13.1. The items marked with “found” in the table represent the number of tokens that were covered by our WP data, i.e. the tokens in the spoken–academic corpora that matched with at least one token in our WP text data. We can remark on a few things:

- Despite greater numbers of documents and 1-gram occurrences, the spoken data was considerably less diverse than the academic data, with only 10,491 unique tokens in comparison to 14,331. The coverage (i.e. percentage of 1-grams in the data that were present in our Wikipedia data) was also greater than what was found in the academic data, meaning that the academic data was lexically more diverse and contained on average less dispersed 1-grams.
- The unique-to-total ratio in terms of the 1-grams that were found once again supports our rationale for not including it in a difficulty value: both the spoken and the academic data show a very similar unique-to-total ratio, with the spoken data being only 2 percent more “diverse” than the academic data. This type of calculation (commonly termed *type-to-token*) is often used as a measure of lexical diversity or richness (Douglas, 2010; Tweedie and Baayen, 1998), but our results show that this is not a very good predictor of lexical diversity.

⁴ <https://old.sketchengine.co.uk/auth/corpora/>. Accessed on 09/17/2018.

Table 13.1: Results of the spoken–academic text comparison.

	Spoken	Academic
Documents	60	40
Unique 1-grams	11,812	17,471
Unique 1-grams found	10,491	14,331
1-gram occurrences	247,030	245,146
1-gram occurrences found	240,134	236,799
Unique-total ratio (found)	4%	6%
Unique-total ratio	5%	7%
1-gram coverage	89%	82%
Median of unique DP values	0.987	0.993
Mean of unique DP values	0.957	0.967
Median of total DP values	0.671	0.812
Mean of total DP values	0.620	0.644
TDV	0.829	0.902

- The greatest difference between the two groups lies in the median of the DP values of the total occurrences: the spoken data shows a 17 percent lower value (0.671) than the academic data (0.812). If one uses the arithmetic mean, the difference (0.620 to 0.644) is negligible, which shows the robustness of the median and that our considerations on total occurrences were not very far off the mark.
- The difference in TDVs (only 0.073 for 0.829 to 0.902) is not as great as one would expect, given the two quite different sources of the data.

13.2 FURTHER DEMONSTRATIONS

13.2.1 *The 475 random Wikipedia articles*

We analyzed the 475 randomly extracted Wikipedia articles we discussed in section 12.2.1, p. 88 for TDVs.

The four easiest articles out of the 475, with a range of TDVs 0.348—0.532, were all biology-related: *Parasophronica strandiella*, *Adesmiella cordipicta*, *Synuchus rufofuscus*, and *Pyrausta draesekei*. Puzzled by how these arcane article titles were ascribed such low TDVs, we went to the original article files to see the original texts. The answer was in their length: the longest, *Adesmiella cordipicta*, had a size of 26 tokens.

As an example, we will cite the “easiest” article in its entirety, alongside its tokens’ DP values in superscript (the formatting for each token is explained in p. 109):

PARASOPHRONICA^{1.0} **STRANDIELLA**^{1.0} is^{0.355}
a^{0.141} **species**^{0.927} of^{0.154} **BEETLE**^{0.994} in^{0.144}
the^{0.108} *family*^{0.708} **CERAMBYCIDAE**^{0.999}. It^{0.324}
was^{0.275} *described*^{0.739} by^{0.214} **BREUNING**^{0.999}
in^{0.144} 1940.
References^{0.941}

Parasophronica strandiella is an example of a Wikipedia “stub”, an article that is often no longer than a dictionary definition. The main factors that lowered the TDV artificially were that the Latin terminology was not found in our database, and since the first version of our algorithm did not take into account tokens that were not found in our data, they did not contribute to the median; in addition, they had a very high unique-to-total ratio (77 to 95 percent), making the calculation of TDV essentially a calculation of the median of the unique tokens’ DP values (which was filled with low-DP values, like the tokens *is*, *a*, *of*, *in*, *the*, *it*, *by*, etc.

This tells us two things: first, that the TDS is quite weak when it analyzes texts with only a few sentences. The arithmetic mean is a better measure in that regard, yielding higher values (0.503—0.581 for the four easiest articles).

Second, instead of not acknowledging the tokens that were not found in our Wikipedia data and for which DP cannot be calculated, we could ascribe them the maximum value of 1.000. We tested that approach on the *Parasophronica* article and found that the DP value would jump from 0.340 to 0.709, which we argue to be a better appraisal of the difficulty of the article than 0.340.

Table 13.2: Difference in TDVs after ascribing the value of 1.000 to tokens not found in the English data.

	Spoken		Academic	
	<i>Before</i>	<i>After</i>	<i>Before</i>	<i>After</i>
Median of unique DP values	0.987	0.991	0.993	0.997
Mean of unique DP values	0.957	0.962	0.967	0.973
Median of total DP values	0.671	0.698	0.812	0.830
Mean of total DP values	0.620	0.631	0.644	0.657
TDV	0.829	0.844	0.902	0.913

With this information of mind, we asked ourselves whether the results of the spoken—academic corpora analysis would be significantly changed if we applied a DP value of 1.000 for “unknown” tokens. The results of that comparison are shown in 13.2.

The difference in TDVs between the two articles remained practically the same (0.069 versus 0.073), which means that this is a measure that would be mainly useful for correcting the TDVs of very short texts.

There is a small disadvantage to this approach, however, if the corpus being used is relatively small: ascribing a value of 1.000 to the unknown tokens would raise all texts’ TDVs more or less in an equal manner, since a good amount of words in a text that the user is analyzing would not be found in the text data. However, since this is an overall increase due to the deficiencies on the source data, it is arguably a small price to pay in order to be able to analyze shorter texts more accurately.

Therefore, we argue that it is best to ascribe a value of 1.000 to an unknown token (i.e. a token that was not present in the corpus or text data) instead of disregarding the token completely.

13.3 SPOKEN—ACADEMIC PORTUGUESE

Here, we experiment the TDS on another language, Portuguese, to see if the same differences in the spoken—academic opposition would hold. In order to do this, we extracted 10,000 Wikipedia Portuguese articles randomly using our automatic Wikipedia Extractor Jupyter Notebook, for a total of 3.8 million tokens; then,

See p. 111, table 14.1 for the overall statistics of the 10,000-article Portuguese WP data, and the Appendix, page 186, table E.2, for the top 100 tokens in that data ordered by frequency.

we built our text data the same way we built the English Wikipedia data (except for allowing solely 1-grams this time); finally, we built, similarly to the Santa Barbara and the academic article corpus, two small collections of text of roughly the same size and compared them.

This comparison will not be of the same quality as the previous one, since the Portuguese WP text data we are working on is significantly smaller. However, this should suffice for showcasing the flexibility of our proposal to other languages and gauging whether the same tendency of the spoken language having lower TDVs applies for a different language.

13.3.1 *The spoken data*

Containing private and public contexts, with similar quantities of monologues, dialogues (involving two people), and conversations (involving three or more people), the informal part of the C-ORAL-BRASIL (COB) is

a spontaneous speech corpus of Brazilian Portuguese, mainly of the *Mineiro* variant with strong emphasis on the metropolitan region of Belo Horizonte (...) composed by 208,130 words, distributed in 139 texts of, in average, 1,500 words each (Raso and Mello, 2012, pp. 55, 60)⁵

Transcription files in .csv were obtained from the project's website⁶.

Having in mind that spoken Portuguese, in particular the Minas Gerais variant captured by the COB, presents over 1,000 speech forms that are different from written Portuguese, e.g. *ea* instead of *ela*, *aque'* instead of *aquele*, *ca* instead of *com a* (Raso and Mello, 2012, p. 289), and that these different speech forms have been transcribed in the COB with fidelity, we should expect a significant amount of these to be absent in our Wikipedia data,

5 Our translation from the original Portuguese: “O *corpus* C-ORAL-BRASIL é um corpus de fala espontânea do português do Brasil, principalmente da diatopia mineira com forte ênfase na região metropolitana de Belo Horizonte (...) constituída por 208.130 palavras, distribuídas em 139 textos de, em média, 1.500 palavras cada um (...)” (Raso and Mello, 2012, pp. 55, 60).

6 <http://150.164.100.23:3000/>. Accessed on 04/09/2018.

i.e. we should expect lower Wikipedia coverage. This is a different situation compared to Santa Barbara, as English seems to vary less from the spoken to the written register in comparison to Portuguese.

Since this considerably larger amount of Portuguese speech forms may raise significantly the TDVs if assigned the maximum value of 1.000, we decided, for this specific case, to revert to our previous practice of disregarding tokens that were not present in the Wikipedia data.

Thus, instead of ascribing 1.000, the maximum value of *DP*, to tokens that are not found in the Wikipedia text data, they will be simply disregarded. We will still provide the number of these unknown tokens, and compare the two scenarios (ascribing or not ascribing 1.000 to unknown tokens), as we did for English.

As the transcriptions for the *COB* are quite consistent, cleaning was easily performed in Microsoft Excel by the “find and replace” function for the few characters used in the transcription (mainly brackets and slashes). Speaker and recording names were removed, as well as sounds and sequences that the transcriber could not distinguish (denoted by a combination of characters such as *yyy*). The Appendix, p. 176, table C.2, shows an example of the cleaning. This clean *COB* data was then uploaded to SketchEngine, with wordlists generated and used for the calculation of *DP* and *TDV* in the same way as the Santa Barbara.

13.3.2 *The academic data*

To search on Scopus, we used a special query of the type SUBJAREA (medi OR nurs OR vete OR dent OR heal OR mult) LANGUAGE (portuguese) AND (LIMIT-TO (ACESSTYPE(OA))) AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "ip")) AND (LIMIT-TO (LANGUAGE , "Portuguese")) .

We built our Portuguese academic article text data following similar guidelines to the academic corpus we used in opposition to Santa Barbara (Kwary, 2018), i.e. by extracting a similar amount of articles in each of the four Scopus categories (Health Sciences, Life sciences, Physical Sciences, and Social Sciences) in order to match the *COB*'s size.

Cleaning is especially challenging for academic articles, and we had to resort to manual extraction of the text files through copying and pasting the desired sections. Similarly to Kwary (2018), we extracted only the title, body of text of the article, tables, and captions for figures, up to and including the final remarks. We did not include the names of the authors, the abstracts (either in English or in Portuguese; as they are mainly repetitions of the con-

Table 13.3: Academic Portuguese data statistics, with the number of tokens estimated by SketchEngine.

	Articles	Tokens
<i>Health sciences</i>	14	36,378
<i>Life sciences</i>	14	42,778
<i>Physical Sciences</i>	13	68,281
<i>Social Sciences</i>	10	61,411
Total	51	208,848

tent of the article in summarized form, they inflate the frequency counts for the words used in the article), and the references, as these sections present plenty of proper names (researchers, article names, journal names, etc.) that could be considered “confounding factors” in terms of difficulty analysis. Unfortunately, it is quite impractical to remove in-text citations of studies. Section names such as introduction, discussion, and so on were kept. Overall information on the academic Portuguese data is shown in table 13.3.

Again we uploaded the text files to SketchEngine and generated the frequency wordlists for lowercase Portuguese words. For the academic data, we continued the practice of assigning the maximum value of 1.000 to the tokens not present in our Wikipedia data.

13.4 ANALYSIS RESULTS

The data related to the spoken—academic comparison for Portuguese is shown in table 13.4. In order to allow for better comparison to the English data, we included the previous English spoken—academic comparison in the table. We also compared disregarding tokens versus assigning to them the maximum value of 1.000, as we did for English, on table 13.5.

We can observe the following:

- 1-gram coverage was lower for both the spoken and academic data in Portuguese, which may indicate two things: first, that the Wikipedia data we used as basis (10,000 Portuguese WP articles versus 124,000 English articles) was less comprehensive in terms

Table 13.4: Results of the spoken–academic text comparison for English and Portuguese.

	English		Portuguese	
	Spoken	Academic	Spoken	Academic
<i>Documents</i>	60	40	139	51
<i>Unique 1-grams</i>	11,812	17,471	12,469	20,882
<i>Unique 1-grams (found)</i>	10,491	14,331	9,567	15,476
<i>1-gram occurrences</i>	247,030	245,146	207,584	208,821
<i>1-gram occurrences (found)</i>	240,134	236,799	195,102	196,514
<i>Unique-total ratio (found)</i>	4%	6%	5%	8%
<i>Unique-total ratio</i>	5%	7%	6%	10%
<i>1-gram coverage</i>	89%	82%	77%	74%
<i>Median of unique DP values</i>	0.987	0.993	0.987	0.996
<i>Mean of unique DP values</i>	0.957	0.967	0.957	0.975
<i>Median of total DP values</i>	0.671	0.812	0.794	0.808
<i>Mean of total DP values</i>	0.620	0.644	0.669	0.649
TDS	0.829	0.902	0.891	0.902

of the lexicon; and second, that the spoken and academic data simply had tokens that are by definition absent from Wikipedia data. Both possibilities are highly likely, considering that the Portuguese spoken data has hundreds of non-orthographic speech forms and that the academic data has specialized terms and jargon that not even a large amount of Wikipedia articles can cover (as evidence of this, notice the low coverage for the academic data for English even with such a large collection of Wikipedia texts).

- We observe a small difference (only 0.012) in TDV comparisons for Portuguese, this time much smaller than in English (0.073). This may be in part caused by the issues of lower coverage we discussed on the previous item and the fact that several speech-exclusive forms such as *es* (*eles*, they) and *nũ* (*nã*o, no / not / do not) that were excluded were especially frequent in the COB, thus raising slightly the TDV.
- The unique DP values are remarkably similar, even identical, for both English and Portuguese comparisons, while we have the greatest differences for the total DP values. While the English data presents a large difference between the median of total DP values

Table 13.5: Difference in TDVs after ascribing the value of 1.000 to tokens not found in the Portuguese data.

	Spoken		Academic	
	<i>Before</i>	<i>After</i>	<i>Before</i>	<i>After</i>
Median of unique DP values	0.987	0.994	0.989	0.996
Mean of unique DP values	0.957	0.967	0.966	0.975
Median of total DP values	0.794	0.821	0.761	0.808
Mean of total DP values	0.669	0.689	0.627	0.649
TDV	0.891	0.908	0.875	0.902

for the spoken and academic data (0.141), the Portuguese data presents a much smaller difference (0.018), indicating that there was a similar distribution of use of low and high DP tokens for both the spoken and academic data. Again this is likely to have been affected by the coverage issues already discussed, especially the nature of each text data.

The difference in TDVs before and after assigning 1.000 to the uncovered tokens, shown in table 13.5, was larger in the Portuguese academic data (0.027), showing that the absent academic tokens were more crucial for the estimation of difficulty than even the Portuguese spoken data, which had a small difference of 0.011 in TDVs.

This shows that for base text data with lower coverage, such as the 10,000-article Wikipedia data, ascribing the maximum value for unknown tokens can yield greater differences before and after than data with higher coverage, as was our English data, and therefore this is a measure that is not solely helpful for shorter texts, as previously stated in section 13.2.1, but also for text data that provides lower coverage.

13.5 SUMMARY

In this chapter, we have attempted to design a Text Difficulty Scale (TDS) that was consistent with the empirical evidence in linguistics. When comparing spoken and academic data, we discovered a smaller than expected difference in TDVs for both English and

Portuguese, which, despite the methodological limitations of each comparison, puts into question the usefulness of our proposal. As stated previously, actual difficulty testing through multiple choice questions and a more representative, varied corpus of text including non-written text sources, such as formal and informal spontaneous speech, could help show the adequacy of the [TDS](#).

In the next chapter, we will explore the Difficulty Highlighter, a Python script that makes a [TDV](#) analysis and outputs the results with formatting dependent on the [DP](#) value of each token.

THE DIFFICULTY HIGHLIGHTER

14.1 DESIGN CONSIDERATIONS

We designed a difficulty highlighter in Python that changes formatting of tokens based on a range of **DP** values. We originally made a five-point scale, with the categories *very easy*, *easy*, *average*, *difficult*, and *very difficult*, but after extensive testing on actual texts we found that including an additional category, *slightly difficult*, reflected better the distribution of **DP**, which is more advantageous since we have three equal groups of two categories each (easy, average, difficult). We thus have:

- Very easy, for tokens in the range 0.000—0.700;
- *Easy*, for tokens in the range 0.701—0.800;
- Average, for tokens in the range 0.801—0.900;
- **Slightly difficult**, for tokens in the range 0.901—0.950;
- **DIFFICULT**, for tokens in the range 0.951—0.990;
- **VERY DIFFICULT**, for tokens in the range 0.991—1.000.

These categories could potentially serve as guidelines for the **TDS**, so that if a text had a **TDV** of over 0.90, it could be considered difficult, for instance. More empirical testing to confirm these assumptions, however, is required, and they would likely be different for each language and text data.

We had only a few options to choose for text formatting. If you use full color capability and many ranges of **DP** values, the text becomes nearly unreadable, with approximate colors (like orange and red) being hard to discern.

Thus, we used blue (ultramarine) and red (a complementary tone to ultramarine), and when deciding the formatting style we opted for a kind of ascending scale, with very easy closed-class words like *the*, *and*, *of* not needing to be highlighted, i.e. our baseline; then, we use italics for the easy words; a semibold variant for

The .pdf version of this thesis contains the difficulty-highlighted sections in full color.

the average words; and for the group of difficult words we use first a bold and blue scheme, then a small caps scheme in black with a lighter weight. Finally, we use a thicker red style in small caps for the really difficult words. There is an alternate scheme for all-black printing which replaces the semibold/bold opposition with an underlined/bold opposition.

For languages that use non-Latin alphabets or writing systems, which do not normally have small caps or italics support, we decided to change the italics for an underline, whereas the small caps were changed to a grey gradient background.

The difficulty highlighter functions as follows: first, it prints the TDV for the text on top; then, the highlighted text; and finally a wordlist with all unique tokens by difficulty category (which can help if one needs to translate a text, for instance). The option to show the DP values as superscript (in a light grey color), as shown further in our examples, can be toggled off quite easily.

In terms of output, the difficulty highlighter can generate highlighted text in both .html, an advantageous file format as it can be easily copied, formatting included, to Microsoft Word or other rich text programs, and .tex, for L^AT_EX. The latter requires some adaptation for non-Latin alphabets that is not included, making the .html the most flexible choice.

In the next sections, we will demonstrate the difficulty highlighter – which is perhaps the best way to visualize how our proposal estimates difficulty – with short texts from different sources.

14.2 JOHANN SEBASTIAN BACH IN FIVE LANGUAGES

We demonstrate our difficulty highlighter by working with summaries from Wikipedia in five different languages: English¹ (using our 124,000-article data), Portuguese² (using the aforementioned 10,000-article data), Japanese³, Hebrew⁴, and Persian⁵.

¹ https://en.wikipedia.org/wiki/Johann_Sebastian_Bach. Accessed on 09/13/2018.

² https://pt.wikipedia.org/wiki/Johann_Sebastian_Bach. Accessed on 09/13/2018.

³ <https://goo.gl/PUd9Zm>. Accessed on 09/13/2018.

⁴ <https://goo.gl/T758HL>. Accessed on 09/13/2018.

⁵ <https://goo.gl/5SWVhE>. Accessed on 09/13/2018.

Table 14.1: Overall statistics for the Wikipedia text data in English, Portuguese, Japanese, Hebrew, and Persian.

	English	Portuguese	Japanese	Hebrew	Persian
Articles	123,974	9,998	7,050	2,336	2,628
1-gram occurrences (in millions)	61.50	3.80	2.19	0.36	0.62
Unique 1-grams	955,472	198,222	146,537	141,336	68,339
Token with lowest DP	the (0.109)	de (0.139, of)	の (0.112, no, syllable)	של (0.153, shel, of)	در (0.095, dar, at)

We assembled text data for each language with our automatic Wikipedia extractor, which obtained random articles. Overall text data for each language is shown in table 14.1. Lists of the top 100 tokens for each language by frequency, with accompanying translations, are in the Appendix, chapter E.

To allow an appraisal of the tokens' difficulty estimations, we also provide automated translations from Portuguese, Hebrew, Japanese, and Persian into English, retaining most of the original difficulty highlighting. These are contained in the Appendix, chapter D.

14.2.1 *English*

Legend — Very easy | *Easy* | Average | **Slightly difficult** | DIFFICULT | **VERY DIFFICULT** |

JOHANN^{0.981} SEBASTIAN^{0.988} **BACH**^{0.992} (31 March^{0.693} [**O.S.**^{0.998} . 21 March^{0.693}] 1685 – 28 July^{0.707} 1750) was^{0.275} a^{0.141} **German**^{0.839} COMPOSER^{0.961} and^{0.112} MUSICIAN^{0.960} of^{0.154} the^{0.108} BAROQUE^{0.986} period^{0.693}. He^{0.593} is^{0.355} known^{0.526} for^{0.221} INSTRUMENTAL^{0.950} COMPOSITIONS^{0.978} such^{0.504} as^{0.200} the^{0.108} **BRANDENBURG**^{0.995} **CONCERTOS**^{0.997} and^{0.112} the^{0.108} **GOLDBERG**^{0.995} **Variations**^{0.947} as^{0.200} well^{0.499} as^{0.200} for^{0.221} VOCAL^{0.959} music^{0.839} such^{0.504} as^{0.200} the^{0.108} **St**^{0.924} MATTHEW^{0.969} PASSION^{0.970} and^{0.112} the^{0.108} Mass^{0.879} in^{0.144} **B**^{0.905} minor^{0.872}. Since^{0.534} the^{0.108} 19TH-CENTURY^{0.978} **BACH**^{0.992} REVIVAL^{0.954} he^{0.593} has^{0.415} been^{0.398} *generally*^{0.756} regarded^{0.874} as^{0.200}

one^{0.351} of^{0.154} the^{0.108} greatest^{0.874} COMPOSERS^{0.981} of^{0.154} all^{0.410} time^{0.438}. **BACH**^{0.992} **ENRICHED**^{0.991} established^{0.719} German^{0.839} styles^{0.945} through^{0.512} his^{0.545} MASTERY^{0.989} of^{0.154} **COUNTERPOINT**^{0.994}, **HARMONIC**^{0.991} and^{0.112} **MOTIVIC**^{0.999} organisation^{0.942}, and^{0.112} his^{0.545} ADAPTA-
 TION^{0.952} of^{0.154} RHYTHMS^{0.989}, forms^{0.839}, and^{0.112} **TEX-
 TURES**^{0.995} from^{0.242} abroad^{0.944}, particularly^{0.770} from^{0.242}
 Italy^{0.895} and^{0.112} France^{0.843}. **BACH**^{0.992}'s^{0.340} COMPOSI-
 TIONS^{0.978} include^{0.669} hundreds^{0.909} of^{0.154} **CANTATAS**^{0.999}
 , both^{0.473} SACRED^{0.959} and^{0.112} SECULAR^{0.966}. He^{0.593} com-
 posed^{0.863} Latin^{0.897} church^{0.850} music^{0.839}, **PASSIONS**^{0.994}
 , **ORATORIOS**^{0.998}, and^{0.112} **MOTETS**^{0.999}. He^{0.593} often^{0.653}
 adopted^{0.842} LUTHERAN^{0.986} **HYMNS**^{0.991}, not^{0.385} only^{0.431}
 in^{0.144} his^{0.545} larger^{0.808} VOCAL^{0.959} works^{0.799}, but^{0.362}
 for^{0.221} instance^{0.906} also^{0.316} in^{0.144} his^{0.545} **FOUR-PART**
 0.997 **CHORALES**^{0.999} and^{0.112} his^{0.545} SACRED^{0.959} songs^{0.914}
 . He^{0.593} wrote^{0.779} extensively^{0.947} for^{0.221} ORGAN^{0.973} and
 0.112 for^{0.221} other^{0.378} KEYBOARD^{0.987} INSTRUMENTS^{0.952}
 . He^{0.593} composed^{0.863} **CONCERTOS**^{0.997}, for^{0.221} instance
 0.906 for^{0.221} VIOLIN^{0.989} and^{0.112} for^{0.221} **HARPSICHORD**
 0.998, and^{0.112} **SUITES**^{0.993}, as^{0.200} chamber^{0.946} music^{0.839}
 as^{0.200} well^{0.499} as^{0.200} for^{0.221} ORCHESTRA^{0.966}. Many^{0.479}
 of^{0.154} his^{0.545} works^{0.799} EMPLOY^{0.965} the^{0.108} GENRES^{0.975}
 of^{0.154} CANON^{0.974} and^{0.112} **FUGUE**^{0.998}. Throughout^{0.706} the
 0.108 18th^{0.915} century^{0.726} **BACH**^{0.992} was^{0.275} mostly^{0.804}
 RENOWNED^{0.960} as^{0.200} an^{0.255} **ORGANIST**^{0.996}, while^{0.461}
 his^{0.545} KEYBOARD^{0.987} music^{0.839}, such^{0.504} as^{0.200} The^{0.108}
 Well^{0.499}-Tempered **CLAVIER**^{0.999}, was^{0.275} APPRECIATED^{0.984}
 for^{0.221} its^{0.431} **DIDACTIC**^{0.997} QUALITIES^{0.968}. The^{0.108} 19th^{0.860}
 century^{0.726} saw^{0.774} the^{0.108} publication^{0.915} of^{0.154}
 some^{0.465} major^{0.642} **BACH**^{0.992} **BIOGRAPHIES**^{0.991}, and^{0.112}
 by^{0.214} the^{0.108} end^{0.594} of^{0.154} that^{0.301} century^{0.726} all^{0.410} of^{0.154}
 his^{0.545} known^{0.526} music^{0.839} had^{0.422} been^{0.398} printed^{0.948}.
DISSEMINATION^{0.991} of^{0.154} SCHOLARSHIP^{0.961} on^{0.266}
 the^{0.108} COMPOSER^{0.961} continued^{0.696} through^{0.512} **PERIOD-
 ICALS**^{0.991} and^{0.112} WEBSITES^{0.980} exclusively^{0.933} devoted^{0.943}
 to^{0.149} him^{0.671}, and^{0.112} other^{0.378} publications^{0.944} such
 0.504 as^{0.200} the^{0.108} **BACH**^{0.992}-Werke-Verzeichnis (**BWV**^{0.999},
 a^{0.141} NUMBERED^{0.966} CATALOGUE^{0.986} of^{0.154} his^{0.545} works^{0.799})
 and^{0.112} new^{0.484} critical^{0.865} EDITIONS^{0.963} of^{0.154} his

0.545 COMPOSITIONS 0.978 . His 0.545 music 0.839 was 0.275 further 0.661 **POPULARISED** 0.994 through 0.512 a 0.141 MULTITUDE 0.989 of 0.154 ARRANGEMENTS 0.958 , including 0.495 for 0.221 **instance** 0.906 the 0.108 Air 0.827 on 0.266 the 0.108 G 0.966 STRING 0.959 , and 0.112 of 0.154 RECORDINGS 0.962 , for 0.221 **instance** 0.906 three 0.508 different 0.664 **box** 0.925 **sets** 0.920 with 0.211 *complete* 0.795 **performances** 0.928 of 0.154 the 0.108 COMPOSER 0.961 's 0.340 *works* 0.799 MARKING 0.962 the 0.108 **250TH** 0.999 **anniversary** 0.926 of 0.154 his 0.545 *death* 0.730 .

14.2.2 Portuguese, Japanese, Hebrew, and Persian

For these four languages, some of which presented difficulties in typesetting in L^AT_EX, we converted the .html difficulty-highlighted versions to .pdf and then to images using Adobe Photoshop. The images were cropped to adjust to one page each, so they show only part of the Difficulty Highlighter .html file. Figs. 14.1, 14.2, 14.3, and 14.4 show the highlighted texts. Have in mind that due to Hebrew and Persian being written right to left, DP values here are depicted to the left, and not to the right, to the respective tokens.

In regards to Japanese, the difficulty-highlighted version is unnaturally segmented for the sake of difficulty analysis; in reality, Japanese texts have few breaks between strings of characters.

While Text Difficulty Values found for English (0.789431), Portuguese (0.785106), and Persian (0.778699) were quite similar, the Text Difficulty Values found for the other two languages were 0.673843 (Japanese) and 0.979787 (Hebrew). This discrepancy between the two groups may be due to the text data sizes, which were quite different, and to the fact that the ranges of DP values for every difficulty category may be different from one language to the other, as each language's morphology and use of the lexicon may differ. For instance, in terms of morphology, English and Portuguese are quite analytical overall; in comparison, Hebrew, Japanese and Persian are more synthetic; and there may be other differences in lexical choices, such as number and frequency of *n*-grams, that have not been controlled for.

JOHANN^{0.982} SEBASTIAN^{0.986} **BACH**^{0.992} (**EISENACH**^{0.999}, 21
 de^{0.139} março^{0.713} de^{0.139} 1685 — **LEIPZIG**^{0.995}, 28 de^{0.139}
 julho^{0.709} de^{0.139} 1750) foi^{0.329} um^{0.239} **compositor**^{0.932},
CRAVISTA^{0.999}, **KAPPELLMEISTER**^{0.999}, REGENTE^{0.961},
ORGANISTA^{0.998}, professor^{0.855}, **VIOLINISTA**^{0.996} e^{0.142}
VIOLISTA^{1.0} **ORIUNDO**^{0.990} do^{0.218} SACRO^{0.985} Império^{0.866}
 ROMANO-GERMÂNICO^{0.985}, atual^{0.752} Alemanha^{0.866}.
 Nascido^{0.898} numa^{0.702} família^{0.740} de^{0.139} longa^{0.868}
 tradição^{0.861} musical^{0.869}, cedo^{0.936} **mostrou**^{0.900}
possuir^{0.917} TALENTO^{0.959} e^{0.142} logo^{0.715} tornou-se^{0.736} um^{0.239}
músico^{0.943} completo^{0.883}. ESTUDANTE^{0.952}
INCANSÁVEL^{0.995}, **adquiriu**^{0.947} um^{0.239} VASTO^{0.965}
 conhecimento^{0.861} da^{0.219} música^{0.802} europeia^{0.893} de^{0.139}
 sua^{0.372} época^{0.659} e^{0.142} das^{0.377} **gerações**^{0.925}
 anteriores^{0.834}. DESEMPENHOU^{0.968} vários^{0.614} **cargos**^{0.948}
 em^{0.187} CORTES^{0.953} e^{0.142} **igrejas**^{0.944} ALEMÃS^{0.977}, mas^{0.434}
 suas^{0.496} funções^{0.897} mais^{0.324} **DESTACADAS**^{0.992} foram^{0.490}
 a^{0.157} de^{0.139} **KANTOR**^{1.0} da^{0.219} Igreja^{0.848} de^{0.139} São^{0.487}
 TOMÁS^{0.973} e^{0.142} Diretor^{0.850} Musical^{0.869} da^{0.219} cidade^{0.681}
 de^{0.139} **LEIPZIG**^{0.995}, onde^{0.497} **desenvolveu**^{0.908} a^{0.157}
 parte^{0.509} final^{0.634} e^{0.142} mais^{0.324} importante^{0.714} de^{0.139}
 sua^{0.372} carreira^{0.796}. **ABSORVENDO**^{0.999} inicialmente^{0.789} o^{0.180}
 grande^{0.507} REPERTÓRIO^{0.966} de^{0.139} música^{0.802}
CONTRAPONTÍSTICA^{1.0} **GERMÂNICA**^{0.993} como^{0.288} base^{0.718}
 de^{0.139} seu^{0.407} estilo^{0.791}, recebeu^{0.731} mais^{0.324} tarde^{0.681} a^{0.157}
 influência^{0.787} italiana^{0.938} e^{0.142} francesa^{0.893}, através^{0.620}
 das^{0.377} quais^{0.715} sua^{0.372} obra^{0.809} se^{0.333} **ENRIQUECEU**^{0.999}

Figure 14.1: *Johann Sebastian Bach* Portuguese WP article highlighted for difficulty.

ヨ^{0.935}ハン^{0.932}・ゼバスティアン^{0.999}・バッハ^{0.987} (JOHANN^{0.998} SEBASTIAN^{0.999} BACH^{0.999}, 1685年^{0.398} 3月^{0.475} 31日^{0.511} (ユリウス^{0.992} 暦^{0.961} 1685年^{0.398} 3月^{0.475} 21日^{0.511}) - 1750年^{0.398} 7月^{0.475} 28日^{0.511}) は^{0.177}、18世紀^{0.838} の^{0.112} ドイツ^{0.867} で^{0.180} 活躍^{0.810} した^{0.281} 作曲家^{0.965} ・音楽家^{0.980} で^{0.180} ある^{0.314}。バロック^{0.994} 音楽^{0.825} の^{0.112} 重要^{0.821} な^{0.259} 作曲家^{0.965} の^{0.112} 一人^{0.776} で^{0.180}、鍵盤^{0.997} 楽器^{0.967} の^{0.112} 演奏家^{0.995} として^{0.303} も^{0.277} 高^{0.688} 名^{0.590} で^{0.180} あり^{0.431}、当時^{0.638} から^{0.258} 即興^{0.988} 演奏^{0.927} の^{0.112} 大家^{0.990} として^{0.303} 知^{0.643} ら^{0.345} れ^{0.247} てい^{0.360} た^{0.239}。バッハ^{0.987} 研究者^{0.942} の^{0.112} 見解^{0.944} では^{0.368}、バッハ^{0.987} は^{0.177} バロック^{0.994} 音楽^{0.825} の^{0.112} 最後^{0.717} 尾^{0.944} に^{0.148} 位置^{0.819} する^{0.292} 作曲家^{0.965} として^{0.303} それ^{0.570} まで^{0.432} の^{0.112} 音楽^{0.825} を^{0.190} 集大成^{0.988} した^{0.281} とも^{0.607} 評価^{0.816} さ^{0.255} れる^{0.426} が^{0.198}、後世^{0.964} に^{0.148} は^{0.177}、西洋^{0.959} 音楽^{0.825} の^{0.112} 基礎^{0.916} を^{0.190} 構築^{0.934} した^{0.281} 作曲家^{0.965} で^{0.180} あり^{0.431} 音楽^{0.825} の^{0.112} 源流^{0.983} で^{0.180} ある^{0.314} とも^{0.607} 捉^{0.958} え^{0.482} ら^{0.345} れ^{0.247}、日本^{0.536} の^{0.112} 音楽^{0.825} 教育^{0.841} では^{0.368} 「音楽^{0.825} の^{0.112} 父^{0.820}」と^{0.185} 称^{0.850} さ^{0.255} れ^{0.247} た^{0.239}。バッハ^{0.987} 一族^{0.925} は^{0.177} 音楽家^{0.980} の^{0.112} 家系^{0.967} で^{0.180} (バッハ^{0.987} 家^{0.680} 参照^{0.783}) 数多く^{0.911} の^{0.112} 音楽家^{0.980} を^{0.190} 輩出^{0.974} した^{0.281} が^{0.198}、中でも^{0.853}、ヨ^{0.935}ハン^{0.932}・ゼバスティアン^{0.999}・バッハ^{0.987} は^{0.177} その^{0.438} 功績^{0.945}

Figure 14.2: *Johann Sebastian Bach* Japanese WP article highlighted for difficulty.

0.952 (בגרמנית) 1.0 **באך** 0.991 **סבסטיאן** 0.990 **יוהאן** 0.998 **JOHANN** 0.998
מידע 0.843 . **עזרה** 0.894 ; (**במרץ** 0.735) 0.999 **SEBASTIAN BACH** 1.0
 – 28 **ביולי** 0.743 (1750) היה 0.429 **מלחין** 0.987 **גרמני** 0.946 **מתקופת** 0.915 1685
 0.998 **הבארוק** 0.990 , **הנמנה** 0.999 עם 0.316 **נגני** 0.995 **העוגב** 0.998
והמלחינים 0.999 **הגדולים** 0.866 ביותר 0.540 **בכל** 0.616 **הזמנים** 0.939 .
 0.995 **יצירותיו** 0.977 של 0.153 **באך** 0.997 , **הכתובות** 0.990 **תוך** 0.692 **מיצוי** 0.995
אמנותי 0.980 **וטכני** 0.996 של 0.153 **טכניקת** 0.993 **הקונטרפונקט** 1.0 ,
נודעות 0.998 **בעומקן** 1.0 **האינטלקטואלי** 0.991 **וביופיין** 1.0
האמנותי 0.982 **ועל** 0.707 **כן** 0.588 **רבים** 0.560 **רואים** 0.945 **בבאך** 1.0 את 0.218
המלחין 0.976 **שהביא** 0.953 **לשיאה** 0.984 את 0.218 **המוזיקה** 0.939
הפוליפונית 0.998 . **בתקופתו** 0.957 **נודע** 0.879 **בעיקר** 0.679 **כנגן** 0.995
עוגב 0.996 **וירטואוז** 0.999 , **ולאחר** 0.739 **מותו** 0.807 **דעכה** 0.993
תהילתו 0.998 , אך 0.476 היא 0.482 **הושבה** 0.993 **לתודעת** 0.995
הציבור 0.840 **במאה** 0.825 ה 0.506 – 19 **ולא** 0.652 **פסה** 1.0 **מאז** 0.716 . בין 0.377
יצירותיו 0.977 **החשובות** 0.944 **והמוכרות** 1.0 ביותר 0.540 : **טוקטה** 0.999
ופוגה 0.999 **ברה** 0.998 **מינור** 0.996 , **הפסנתר** 0.994 **המושווה** 0.999 ,
וריאציות 0.995 **גולדברג** 0.975 , **מתאוס** 0.998 **פסיון** 1.0 , **יוהנס** 0.998
פסיון 1.0 , **הקונצ'רטי** 0.999 **הברנדבורגים** 1.0 , **המנחה** 0.969
המוזיקלית 0.989 , **אמנות** 0.934 **הפוגה** 0.987 , **המיסה** 0.996 **בסי** 0.997
מינור 0.996 **ועוד** 0.777 . **יצירותיו** 0.977 של 0.153 **באך** 0.997 **זכו** 0.871 **וזוכות** 1.0
עד 0.433 **היום** 0.695 **לפופולריות** 0.978 **רבה** 0.766 **מאוד** 0.714 **והוא** 0.650 **נחשב** 0.808
לדעת 0.908 **רבים** 0.560 **לאחד** 0.846 **המלחינים** 0.998 **הגדולים** 0.866
 0.883 " . **בהיסטוריה**

Figure 14.3: *Johann Sebastian Bach* Hebrew WP article highlighted for difficulty.

0.999 **JOHANN** : یوهان 0.960 **سیباستیان** 0.999 **پاخ** (به 0.115 آلمانی 0.776 ،
 زاده 21 مارس 0.805 1685 - درگذشته 28) (**SEBASTIAN BACH**)
 0.961 و 0.114 نوازنده ارگ 0.878 و 0.114 **هارپسیکورد** 1.0
 اهل 0.780 آلمان 0.831 بود 0.342 که 0.138 آثارش 0.965 برای 0.275 دسته آواز 0.942 ،
ارکستر 0.994 و 0.114 تک 0.839 **سازها** 0.944 ، تقریباً تمام 0.509 انواع 0.729
متفاوت 0.762 **موسیقی** 0.797 دوره **باروک** 0.995 را 0.222 **در برمی** 1.0 گیرد 0.734
 و 0.114 **موسیقی** 0.797 آن 0.223 دوران 0.470 را 0.222 به 0.113 **بلوغ** 0.933 **خویش** 0.726
رسانده 0.984 است 0.274 . با 0.169 وجود 0.409 اینکه 0.493 او 0.527 شکل 0.545
تازه 0.710 ای 0.683 را 0.222 از 0.105 **موسیقی** 0.797 ارائه 0.592 **نکرد** 0.773 ، ولی 0.456
 توانست 0.633 **سبک شایع** 0.974 در 0.095 **موسیقی** 0.797 آلمان 0.831 را 0.222 با 0.169
فنون 0.971 **کنترل پوان** 1.0 ، **غنی** 0.895 **سازد** 0.871 . این 0.168 **فنون** 0.971 ، **مینی** 0.700
 بر 0.236 **تنظیم** 0.827 **چیدمان** 0.997 **هارمونیک** 0.999 و 0.114 **موتیف** 0.999
 در 0.095 **مقیاس** 0.927 کوچک 0.599 و 0.114 بزرگ 0.423 بود 0.342 و 0.114
 همچنین 0.346 **سازواری** 0.998 **ریتم** 0.998 **ها** 0.919 و 0.114 **بافت** 0.802 **های** 0.940
یرگرفته 0.736 از 0.105 **موسیقی** 0.797 کشورهای 0.604 دیگر 0.286 چون 0.483
ایتالیا 0.830 و 0.114 فرانسه 0.688 را 0.222 شامل 0.497 **می** 0.875 شد 0.304 **لطفات** 1.0
قوی 0.738 **موسیقی** 0.797 **پاخ** 1.0 و 0.114 **گسترده** 0.935 حاصل 0.673 هنر 0.664
 وی 0.597 ، او 0.527 را 0.222 به 0.113 عنوان 0.359 یکی 0.338 از 0.105 بزرگ 0.423
ترین 0.994 آهنگسازان 0.956 غرب 0.545 در 0.095 **سنت** 0.764 **ثنال** 1.0 مشهور 0.643
 ساخته 0.591 است 0.274 . با 0.169 توجه 0.434 به 0.113 **عمق** 0.857 **معنایی** 0.986 ،
قوت 1.0 **فنی** 0.705 و 0.114 **زیبایی** 0.887 هنر مندانه 0.989 ، **آثاری** 0.856 چون 0.483
کنسرتوهای 0.997 **براندنبورگ** 1.0 ، **سونیت** 0.999 **ها** 0.919 و 0.114
پارتیتاها 1.0 برای 0.275 ساز 0.857 **کلاویه** 1.0 ای 0.683 ، **مس** 0.915 در 0.095
 سی 0.684 **میزور** 0.999 ، **پاسیون** 1.0 به 0.113 روایت 0.693 **متای** 1.0 **قدیس** 0.966 ،
پیشکش 0.978 **موسیقایی** 0.948 ، هنر 0.664 **فوک** 1.0 و 0.114 تعداد 0.560 فراوانی 0.642
کانتات 1.0 که 0.138 **۲۲۰ تا** 1.0 آن 0.105 از 0.223 **ها** 0.919 **برجا** 0.980 **ماده** 0.819
 ' . است 0.274 ، را 0.222 **می** 0.875 **توان** 0.799 نام 0.368 برد 0.576

Figure 14.4: *Johann Sebastian Bach* Persian WP article highlighted for difficulty.

In addition, each language version of Wikipedia very likely has a unique composition of articles, with some subjects being over-represented and others underrepresented; only by using a perfectly parallel corpus with translations (these Wikipedia summaries are not translations of each other) one could perhaps expect more similar TDVs.

Notice that the smaller the text data, the higher the number of tokens found in the “difficult” and “very difficult” categories in the analyzed texts, which points to deficiencies in coverage. This is quite evident if we compare the highlighted versions for English and Portuguese, with the latter having significantly more difficult tokens. Thus, good token coverage, i.e. large and diverse text data as a basis, seems quite important for obtaining good results.

FINAL REMARKS

There is much that needs exploring and testing in the field of text difficulty. This work is but a tentative first step towards a more flexible method of text difficulty analysis. Here, our intention was to be more grounded in the empirical linguistics research in comparison to the other attempts; however, a great deal of guesswork, testing, and re-testing was involved, which is inevitable due to the lack of reliable studies on actual text difficulty. One such example is our initial inclusion of n -gram data, which did not prove to be as important as we thought in actual testing.

Dispersion may have been ideal for this particular project, but it bears mentioning that there are two significant disadvantages in using DP or any other measure of dispersion that takes corpus parts into consideration.

The first is that corpus building is more difficult: ideally, one needs to find many different sources with different genres, contexts, and subjects. Selecting a thousand short stories for inclusion is more difficult than selecting a hundred long novels. Achieving at least a modicum of representativeness of a language is paramount, since the quality of the analysis seems to “live and die” in the quality of the source text data.

Secondly—and we have learned this the hard way—is that even though DP is described as a simple measure, writing a corpus management tool that can account for it is significantly more complex than a corpus management tool that simply counts frequency. There is a great deal of additional information involved—namely, the data on how many tokens there are in every text, and how many texts a given token appeared in— and ultimately one needs to store all that information on disk, so that it is readily available, instead of having to perform the DP calculations every time. For a reasonably large amount of texts (as was our case), this type of DP-enabled program architecture is more demanding on disk and computer resources, taking hours to process, and one needs to know how to program around it.

Other than the real possibility that our proposal for text difficulty analysis, the Text Difficulty Scale, is simply ineffective and ill-construed, we suspect that our results were constrained somewhat by the nature of our text data. Wikipedia, despite its great benefits in terms of ease of data extraction, lexical diversity, and size, seems to be more representative of world knowledge than of language use: articles are written not as spontaneous or “natural” language use, but rather as a store of knowledge that is confined to certain conventions and bounds. This could be ameliorated by including other sources for the text data, such as Internet discussion forums (Reddit¹ is a particularly attractive option we wish to explore in the future) or even YouTube video comments.

We also should have explored lemmas as an alternative option to tokens. Even though there may be different frequencies of use and perhaps even differing dispersions of word forms, lemmas may be more advantageous for a learner and could give better difficulty estimates. On the other hand, it would make the process of text difficulty analysis and difficulty highlighting more complex and harder to implement.

As a side note, one the unintended benefits of this project, unrelated to its original goal of analyzing texts for difficulty, is that it is also a corpus management solution written in Python and reliant on MongoDB that can calculate DP, which the same time is completely customizable to dozens of languages. Ready-made, open source solutions like the one we are offering are rare or even nonexistent.

Going forward, we aim to use this project as a foundation to:

1. Investigate, through extensive, gradual testing, the minimum number of tokens and words necessary for a collection of texts to be sufficiently comprehensive, i.e. for it to get sufficient coverage of texts for analysis. There should be a certain threshold above which little gain in coverage and comprehensiveness is expected, which could save time and effort in building text data. Large corpora are not always better than small corpora, as stated by Brysbaert and New (2009);

¹ <https://old.reddit.com/>. Accessed on 09/29/2018.

2. Design an easy-to-use website that can calculate text difficulty values for several languages, creating text rankings by difficulty. With Likert scales for users' perceived difficulty, and perhaps automated or semi-automated multiple choice tests, one could test the validity of our proposal in a large scale, provided there is enough user interest;
3. Change text difficulty values according to user's vocabulary test results, so that someone who is quite proficient in a language will get lower difficulty values overall than someone who is a beginner;
4. As a more precise alternative to the previous item, design a vocabulary tracker, where users can track the words they already learned, which are then excluded from difficulty analyses;
5. Design a browser extension that can highlight text difficulty of web pages instantaneously;
6. Create separate databases with similar amounts of text data for movies, novels, TV shows, song lyrics, videogames, in order to see the requirements and differing TDVs for each.

Part IV

APPENDIX



EXTRACTING WIKIPEDIA DUMP FILES

The following code was run in Jupyter Notebook for Anaconda with a Python 3.6 distribution¹, in a Dell 15 Gaming notebook with an Intel i7-7700HQ 2.8 GHz CPU and 16 GB RAM.

A.1 WORKING WITH THE WIKIPEDIA DUMP

A.1.1 *Extracting the Wikipedia dump into separate files*

In order to speed up the process, we made a duplicate of the code below (as a separate Python 3 Jupyter Notebook) and ran it simultaneously, the first for half of the subfolders and the second for the other half. We found that running more than two concurrent Python notebooks was too taxing on our system's memory.

Listing A.1: Extracting subdirectories of WP dump into text files

```
import re
import collections
import os
from os import listdir # for getting texts
from os.path import isfile, join

def loadsubfolders(folder_path):
    '''Returns a list of subdirectories in a given path.
    ...
    return [f.path for f in os.scandir(folder_path) if f
            .is_dir()]

def loadfilenames(path):
    '''Returns a list of filenames in a given path.'''
    return [f for f in listdir(path) if isfile(join(path
        , f))]

def opentextfile(path, filename, chosen_encoding):
```

¹ <https://www.anaconda.com/download/>. Accessed on 05/15/2018.

```

        with open(os.path.join(path, filename), encoding=
            chosen_encoding) as t:
            return t.read()

def writetofile(path, filename, text_as_string):
    t = open(os.path.join(path, filename), 'w', encoding
        ='utf-8')
    t.write(text_as_string)
    t.close()

def checkexists(path, filename):
    if filename in [f for f in listdir(path) if isfile(
        join(path, f))]:
        return True
    else:
        return False

folder_path = "F:\\Wikipedia Dump"

subfolders = loadsubfolders(folder_path)

for subfolder in subfolders:

    print('\n\n', subfolder)
    extracted_subfolder = subfolder + '\\extracted'
    extracted_subfolder_contents = [f for f in listdir(
        extracted_subfolder) if isfile(join(
            extracted_subfolder, f))]
    filenames = loadfilenames(subfolder)
    for filename in filenames:
        print(filename, end=', ')
        text_as_string = opentextfile(subfolder,
            filename, 'utf-8')
        text_as_list = text_as_string.split('</doc>\n')
        for article in text_as_list:
            article_title = re.findall(r'title=\\"(.*)\\"",
                article)
            if article_title != []:
                output_filename = article_title[0] + '.
                    txt'
                output_filename_sanitized = "".join( x
                    for x in output_filename if (x.
                        isalnum() or x in "._- "))

```

```

if output_filename_sanitized in
    extracted_subfolder_contents:
    True
else:
    if output_filename_sanitized == 'AUX
        .txt' or
        output_filename_sanitized == '
        aux.txt' or
        output_filename_sanitized == '
        Aux.txt':
        output_filename_sanitized = '
            aux_altered.txt'
    elif output_filename_sanitized == '
        PRN.txt' or
        output_filename_sanitized == '
        prn.txt' or
        output_filename_sanitized == '
        Prn.txt':
        output_filename_sanitized = '
            prn_altered.txt'
    writetofile(extracted_subfolder,
        output_filename_sanitized,
        article)

```

Windows does not allow users to create files named either `aux.txt` or `prn.txt`. This is why we have “sanitized” the filenames for these instances.

A.1.2 *Choosing text files randomly*

With the articles all extracted into separate text files inside the 124 original subfolders, we had to randomly sample each subfolder for around 1,000 texts each, writing the paths to the chosen ones into a `.txt` file. This was accomplished with the code below, again run in a Python Jupiter notebook. This took around two hours.

Listing A.2: Choosing text files randomly

```

import glob
import os
import random

def loadsubfolders(folder_path):

```

```

    '''Returns a list of subdirectories in a given path.
    ...
    return [f.path for f in os.scandir(folder_path) if f
            .is_dir()]

wikisubfolders = loadsubfolders('F:\\Wikipedia Dump\\')

chosen_text_files = []

for subfolder in wikisubfolders:
    choices = random.sample(glob.glob(subfolder+'\\
        extracted\\*.txt'), 1000)
    chosen_text_files.extend(choices)
    print('subfolder', subfolder, 'complete')

with open('.\\AssessingTextDifficulty\\chosen_articles.
    txt', 'w', encoding='utf-8') as l:
    for chosen_text_file in chosen_text_files:
        l.write(chosen_text_file+'\n')

```

The complete list of the selected files can be found here². As these were obtained from text files, which have certain rules for filenames, they can be slightly different from the actual WP article title.

A.2 TEXT DATA

The large size of the text data used in this project (over 10 GB for the English Wikipedia MongoDB database alone), as well as SketchEngine's cap on 30 days for free trials, makes them difficult to share over the Internet, but if you are interested in replicating our results, send us an e-mail over at rubini@ufmg.br that we will make those available to you through a torrent file or other means of your preference.

² <https://www.dropbox.com/s/k6vvbs0r1a11ze8/List-of-randomly-chosen-Wikipedia-articles.txt?dl=0>

B

THE TEXT DIFFICULTY ANALYZER CODE

The following code, in the form of Jupyter Notebooks, is available to download in the project's GitHub page with an MIT license: <https://github.com/filiperubini/text-difficulty-analyzer>.

B.1 AUTOMATIC WIKIPEDIA EXTRACTOR

Listing B.1: Automatic Wikipedia extractor

```
from datetime import datetime, date, time
import os, os.path
import sys

try:
    from tqdm import tqdm_notebook
except:
    try:
        get_ipython().system('conda install --yes --
            prefix {sys.prefix} tqdm')
    except:
        get_ipython().system('{sys.executable} -m pip
            install tqdm')

try:
    import wikipedia
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install wikipedia')

import wikipedia
from tqdm import tqdm_notebook

# Select the language of the articles you want
# and the target number of articles you want

tokenization_language = 'persian'
```

```

target = 100

langs_and_abbr = {'czech': 'cz', 'danish': 'da', 'dutch'
                  : 'nl',
                  'english': 'en', 'estonian': 'et',
                  'finnish': 'fi',
                  'french': 'fr', 'german': 'de', 'greek': 'el',
                  'italian': 'it', 'norwegian': 'no',
                  'polish': 'pl',
                  'portuguese': 'pt', 'slovene': 'sl',
                  'hebrew': 'he',
                  'spanish': 'es', 'swedish': 'sv',
                  'turkish': 'tr',
                  'japanese': 'ja', 'persian': 'fa', 'farsi': 'fa', 'tibetan': 'bo'}

wikipedia.set_lang(langs_and_abbr[tokenization_language
])

get_ipython().system('mkdir output_dir')

# Account for how many text files of the same language
# have already been extracted
existing_files = next(os.walk('./output_dir'))[2]

already_exported = 0

for existing_file in tqdm_notebook(existing_files):
    if existing_file[0:2] == langs_and_abbr[
        tokenization_language]:
        already_exported += 1

remaining = target - already_exported

n_articles = already_exported

print(n_articles, 'articles already exported, with',
      remaining, 'remaining')

# Get random article titles

final_list_of_articles = []

```

```

while remaining != 0:

    if remaining >= 10:
        ten_random_articles = wikipedia.random(pages=10)
        for article in ten_random_articles:
            if article not in final_list_of_articles:
                final_list_of_articles.append(article)
                remaining = remaining - 1
            else:
                pass

    elif remaining < 10:
        for n in range(remaining):
            one_random_article = wikipedia.random(pages
                =1)
            if one_random_article not in
                final_list_of_articles:
                final_list_of_articles.append(
                    one_random_article)
                remaining = remaining - 1
            else:
                pass

# Process and extract articles to text files

print(datetime.now(), 'Wikipedia extraction process
    started for', target, 'files')
for article_title in tqdm_notebook(
    final_list_of_articles):

    try:
        article = wikipedia.page(article_title)
    except:
        final_list_of_articles.remove(article_title)
        print(datetime.now(), 'error when extracting
            article', article_title)

        new_article = wikipedia.random(pages=1)

        while new_article in final_list_of_articles ==
            True:
            new_article = wikipedia.random(pages=1)

        final_list_of_articles.append(new_article)

```

```

        pass

    n_articles += 1

    output_filename = langs_and_abbr[
        tokenization_language] + '{:06}'.format(
        n_articles) + '.txt'

    with open(os.path.join('./output_dir',
        output_filename), 'w', encoding='utf-8') as t:
        t.write(article.content)

print(datetime.now(), 'Wikipedia extraction process
    finished for', target, 'files')

```

B.2 THE CORPUS BUILDER

Listing B.2: The corpus builder

```

# Here, we will build a corpus using Pymongo as a
# database.
# It will install automatically the missing libraries.

# You only need to run this notebook once.

from nltk import sent_tokenize, TreebankWordTokenizer,
    ngrams, WhitespaceTokenizer
from itertools import accumulate, tee, chain
from collections import Counter
import itertools
from datetime import datetime, date, time
from multiprocessing.dummy import Pool as ThreadPool
from collections import defaultdict, OrderedDict
import os, os.path
import re
import string
from decimal import Decimal
from decimal import *
getcontext().prec = 6
import statistics
from IPython.display import display, Markdown, Latex

```

```

# Install necessary libraries if they're not available
import sys

try:
    from tqdm import tqdm_notebook
except:
    try:
        get_ipython().system('conda install --yes --
            prefix {sys.prefix} tqdm')
    except:
        get_ipython().system('{sys.executable} -m pip
            install tqdm')

try:
    import pymongo
    from pymongo import MongoClient
except:
    try:
        get_ipython().system('conda install --yes --
            prefix {sys.prefix} pymongo')
    except:
        get_ipython().system('{sys.executable} -m pip
            install pymongo')

# # Overall Settings

tokenization_language = 'persian' # check the NLTK
    sentence tokenizers for available languages
non_latin_alphabet = True # if your language uses a non-
    Latin alphabet, change to True
files_dir_or_wiki = 'wiki' # change to 'list' if you
    have a list of filenames
    # or to 'dir' if you have a
    specific directory

token_database = 'tokens-persian'
exclude_numbers = True # True / False
exclude_numbers_for_ngrams = False # True / False
number_of_ngrams = 1 # choose up to x number of n-grams
    to extract (minimum = 1)
threads = 4 # decrease this number if you have an old/
    low core processor (minimum = 1)
chosen_encoding = 'utf-8-sig' # better utf-8-sig than
    utf-8, saves trouble

```

```

if non_latin_alphabet == False:
    tokenizer = TreebankWordTokenizer()

elif non_latin_alphabet == True:

    try:
        from polyglot.downloader import downloader
    except ModuleNotFoundError:
        get_ipython().system('{sys.executable} -m pip
            install polyglot')

    try:
        import PyICU, icu
    except ModuleNotFoundError:
        get_ipython().system('{sys.executable} -m pip
            install PyICU')

    try:
        import pycld2
    except ModuleNotFoundError:
        get_ipython().system('{sys.executable} -m pip
            install pycld2 ')

# Installing language-specific module

from polyglot.downloader import downloader

# List of ICU-supported languages

langs = {'afrikaans': 'af', 'alemannic': 'als', 'amharic
': 'am', 'aragonese': 'an', 'arabic': 'ar', '
egyptian arabic': 'arz', 'assamese': 'as', 'asturian
': 'ast', 'azerbaijani': 'az', 'bashkir': 'ba', '
bavarian': 'bar', 'belarusian': 'be', 'bulgarian': '
bg', 'bangla': 'bn', 'tibetan': 'bo', 'bishnupriya':
'bpy', 'breton': 'br', 'bosnian': 'bs', 'catalan':
'ca', 'chechen': 'ce', 'cebuano': 'ceb', 'czech': '
cs', 'chuvash': 'cv', 'welsh': 'cy', 'danish': 'da',
'german': 'de', 'zazaki': 'diq', 'divehi': 'dv', '
greek': 'el', 'english': 'en', 'esperanto': 'eo', '
spanish': 'es', 'estonian': 'et', 'basque': 'eu', '
persian': 'fa', 'finnish': 'fi', 'faroese': 'fo', '
french': 'fr', 'western frisian': 'fy', 'irish': 'ga

```

```

', 'gan chinese': 'gan', 'scottish gaelic': 'gd', '
galician': 'gl', 'gujarati': 'gu', 'manx': 'gv', '
hebrew': 'he', 'hindi': 'hi', 'fiji hindi': 'hif', '
croatian': 'hr', 'upper sorbian': 'hsb', 'haitian
creole': 'ht', 'hungarian': 'hu', 'armenian': 'hy',
'interlingua': 'ia', 'indonesian': 'id', 'iloko': '
ilo', 'ido': 'io', 'icelandic': 'is', 'italian': 'it
', 'japanese': 'ja', 'javanese': 'jv', 'georgian': '
ka', 'kazakh': 'kk', 'khmer': 'km', 'kannada': 'kn',
'korean': 'ko', 'kurdish': 'ku', 'kyrgyz': 'ky', '
latin': 'la', 'luxembourgish': 'lb', 'limburgish': '
li', 'lombard': 'lmo', 'lithuanian': 'lt', 'latvian'
: 'lv', 'malagasy': 'mg', 'macedonian': 'mk', '
malayalam': 'ml', 'mongolian': 'mn', 'marathi': 'mr'
, 'malay': 'ms', 'maltese': 'mt', 'burmese': 'my', '
nepali': 'ne', 'dutch': 'nl', 'norwegian nynorsk': '
nn', 'norwegian': 'no', 'occitan': 'oc', 'odia': 'or
', 'ossetic': 'os', 'punjabi': 'pa', 'pampanga': '
pam', 'polish': 'pl', 'piedmontese': 'pms', 'pashto'
: 'ps', 'portuguese': 'pt', 'quechua': 'qu', '
romansh': 'rm', 'romanian': 'ro', 'russian': 'ru', '
sanskrit': 'sa', 'sakha': 'sah', 'sicilian': 'scn',
'scots': 'sco', 'northern sami': 'se', 'serbo-
croatian': 'sh', 'sinhala': 'si', 'slovak': 'sk', '
slovenian': 'sl', 'albanian': 'sq', 'serbian': 'sr',
'sundanese': 'su', 'swedish': 'sv', 'swahili': 'sw'
, 'silesian': 'szl', 'tamil': 'ta', 'telugu': 'te',
'tajik': 'tg', 'thai': 'th', 'turkmen': 'tk', '
tagalog': 'tl', 'turkish': 'tr', 'tatar': 'tt', '
uyghur': 'ug', 'ukrainian': 'uk', 'urdu': 'ur', '
uzbek': 'uz', 'venetian': 'vec', 'vietnamese': 'vi',
'west flemish': 'vls', 'volapük': 'vo', 'walloon':
'wa', 'waray': 'war', 'yiddish': 'yi', 'yoruba': 'yo
', 'chinese': 'zh', 'chinese character': 'zhc', '
chinese word': 'zhw'}

```

```

if tokenization_language.lower() in langs:

```

```

    token_models = []

```

```

    for lang in langs:

```

```

        code = langs[lang]

```

```

            if tokenization_language in lang:

```

```

        token_models.append(code)

    print(tokenization_language.capitalize(), 'supported
        with models:', token_models)

elif tokenization_language.lower() not in langs:
    print(tokenization_language.capitalize(), 'not
        supported. Reverting to Treebank')
    tokenizer = TreebankWordTokenizer()

# Downloading polyglot language models for the language
# you've chosen

for model in token_models:

    downloader.download('LANG:'+model)

try:
    from polyglot.text import Text
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install morfessor')

from polyglot.text import Text

def icu_tokenizer(text_as_string):

    global token_models

    text = Text(text_as_string)
    text.language = token_models[0]

    return [str(word) for word in text.words]

def icu_sentence_tokenizer(text_as_string):

    global token_models

    text = Text(text_as_string)
    text.language = token_models[0]

    return [str(sentence) for sentence in text.sentences
        ]

```



```

# Execute the mongod file before running next cell.

# Token statistics
client = MongoClient()
db = client[token_database]

token_stats = db[token_database]
token_stats.allowDiskUse=True

# Try getting our text data (if it already exists)
try:
    text_stats = token_stats.find_one({'_text-stats':
        True})['text_stats']
except:
    text_stats = {}

# Delete DB and Indexes, if anything goes wrong
#token_stats.drop()

# Create indexes for faster updates / retrievals
for value in ['token', 'disp', 'freq', 'len', '
    occurred_in', 'dp', '_text-stats']:
    index = token_stats.create_index([(value, pymongo.
        ASCENDING)])

index_occurred_len = token_stats.create_index([('
    occurred_in', pymongo.ASCENDING),
        ('len', pymongo.ASCENDING)])

# # Prerequisite functions

def is_number_repl_isdigit(s):
    '''Returns True is string is a number,
    i.e. if it contains dot or comma.'''

    return re.sub('[.,]', '', s).isdigit()

def cleantokensfromsentence(tokens, bar_numbers):
    '''Clean a list of tokens to remove
    extraneous characters and numerals.
    Bar_numbers: if True, numbers will be deleted'''

```

```

clean_tokens = []

for token in tokens:

    if is_number_repl_isdigit(token) == True:
        if bar_numbers == True:
            '''Barring numbers'''
            pass
        elif bar_numbers == False:
            clean_tokens.append(token)
    elif token.isalnum() is False:
        if len(token) == 1:
            pass
        else:
            matches = 0
            for character in token:
                if character.isalnum() is False:
                    matches += 1
            if matches == len(token):
                pass
            else:
                clean_tokens.append(token)
    else:
        clean_tokens.append(token)

return clean_tokens

def getmultitokens(sentence, n_of_ngrams):
    '''From a list of tokens, generate up to n n-grams
    ...

all_ngrams = []

for x in range(2, n_of_ngrams+1):
    all_ngrams.extend(ngrams(sentence, x))

return [' '.join(i) for i in list(chain(all_ngrams))
        ]

def opentextfile(text_filename):
    global chosen_encoding
    with open(text_filename, 'r', encoding=
              chosen_encoding) as t:
        return t.read()

```

```

def addtodict(key, value, data_dict):
    '''Update a dictionary with a key and value.
    If key exists, combines the values.'''

    if type(value) == int:
        # Sums existing value to current value
        new_value = data_dict.get(key, 0) + value

    elif type(value) == str:
        # Keep existing value
        new_value = data_dict.get(key, value)

    elif type(value) == dict:
        new_value = data_dict.get(key, {})

        for d1_key in value:
            addtodict(d1_key, value[d1_key], new_value)

    data_dict[key] = new_value

def process_sentence(sentence):

    global number_of_ngrams, non_latin_alphabet
    global exclude_numbers, exclude_numbers_for_ngrams

    tempdict = {}

    # Barring numbers on single tokens

    if non_latin_alphabet == False:
        original_tokens = tokenizer.tokenize(sentence)
    elif non_latin_alphabet == True:
        original_tokens = icu_tokenizer(sentence)

    for token in cleantokensfromsentence(original_tokens
    ,exclude_numbers):
        addtodict(token, {'token': token,
            'freq': 1}, tempdict)

    if number_of_ngrams > 1:

```

```

        for token in getmultitokens(
            cleantokensfromsentence(original_tokens,
            exclude_numbers_for_ngrams),
            number_of_ngrams):
            addtodict(token, {'token': token,
                              'freq': 1}, tempdict)

    del original_tokens

    return tempdict

def textdata(text_filename):

    global tokenization_language, non_latin_alphabet

    token_data = {}

    text_as_string = opentextfile(text_filename)

    # Convert to lowercase

    textlc = text_as_string.lower()
    del text_as_string

    # Uncomment this next line of code
    # if you are working with a Wikipedia dump file

    #textlc = textlc.split('>\n')[1]

    results = []

    if non_latin_alphabet == False:
        sentences = sent_tokenize(textlc, language=
            tokenization_language)
    elif non_latin_alphabet == True:
        sentences = icu_sentence_tokenizer(textlc)

    del textlc

    for sentence in sentences:
        results.append(process_sentence(sentence))

    for tempdict in results:

```

```

    for key in tempdict:
        addtodict(key, tempdict[key], token_data)

    return list(token_data.values())

def update_token(token_entry):

    sub_entry = 'freq_in_file' + '.' + str(token_entry['
        file_id'])

    result = token_stats.update_one({'token':
        token_entry['token']},
        {'$inc': {'freq': token_entry['freq'
            ],
            'disp': 1},
        '$setOnInsert': {'len': len(
            token_entry['token'].split(' '))
            },
        '$push': {'occurred_in': str(
            token_entry['file_id']),
            'freq_occurred_in': token_entry
                ['freq']}},
        upsert=True)

def update_db(file_id):

    global token_stats, file_dict

    text_filename = file_dict[file_id]

    token_entries = textdata(text_filename)

    for token_entry in token_entries:
        token_entry['file_id'] = file_id

    pool = ThreadPool(threads)
    results = pool.map(update_token, token_entries)
    pool.close()

def get_text_freq(file_id):

    global text_stats, token_stats, n_of_text_files,
        step, ngramrange

```

```

for n_of_ngrams in ngramrange:

    text_freq = token_stats.aggregate([{'$match':
        {'occurred_in': file_id,
         'len': n_of_ngrams}},
        {'$group': {'_id': None,

                    'freq':
                    {'$sum': {'$arrayElemAt':
                        ['$freq_occurred_in',
                         {'$indexOfArray':
                            ['$occurred_in', file_id
                             ]}}}}}}])

    for result in text_freq:
        text_stats['total'][str(n_of_ngrams)+'-grams
            '][file_id] = result['freq']

#     n_of_text_files += 1

#     if n_of_text_files % step == 0:
#         print(datetime.now(), n_of_text_files, 'text
#             files processed')

# <h1>Text file location</h1>

if files_dir_or_wiki == 'list':
    # Change to the list of filenames of your choice
    list_of_files_path = 'chosen_articles.txt'
    text_files = open(list_of_files_path, 'r', encoding=
        chosen_encoding)
    text_filenames = [x[:-1] for x in text_files.
        readlines()]

elif files_dir_or_wiki == 'dir':
    # Change to the text file directory of your choice
    text_file_directory = './chosen_articles/'
    text_filenames = [os.path.join(text_file_directory, f
        .name) for f in os.scandir(text_file_directory)
        if f.is_file()]

elif files_dir_or_wiki == 'wiki':
    # Point to Wikipedia extracted article directory

```

```

text_file_directory = './output_dir/'
text_filenames = [os.path.join(text_file_directory, f
    .name) for f in os.scandir(text_file_directory)
    if f.is_file()]

# Create a dictionary of filenames
# in order to generate number-filename "code"

f_n = 0
file_dict = {}
for filename in text_filenames:
    file_dict[str(f_n)] = filename
    f_n += 1
del text_filenames

# Calculating corpus parts

if number_of_ngrams <= 1:
    ngramrange = [1]
elif number_of_ngrams > 1:
    ngramrange = range(1, number_of_ngrams+1)

# Here, you will build your corpus by extracting basic
# data
# (frequency, dispersion, the frequency of tokens in a
# given text, etc.)

already_processed = token_stats.distinct('occurred_in')
print(len(already_processed), 'files already processed')

for file_id in already_processed:
    del file_dict[file_id]

n_files = 0
step = ((len(file_dict)) // 100) + 1
half = len(file_dict) // 2
print(len(file_dict), 'files remaining with a step of',
    step, ', half:', half)

try:
    print(datetime.now(), 'started')
    for file_id in tqdm_notebook(list(file_dict.keys()))
        :

```

```

        update_db(file_id)
    print(datetime.now(), 'ended')
except LookupError:
    import nltk
    nltk.download('punkt')

# <h1>Obtain Text Data</h1>

# A prerequisite for calculating DP in the next Part.

# Calculating corpus parts

file_ids = token_stats.distinct('occurred_in')

text_stats = {'total': {}}

for key in tqdm_notebook(text_stats):

    for n_of_ngrams in tqdm_notebook(ngramrange):
        text_stats[key][str(n_of_ngrams)+'-grams'] = {}

n_of_text_files = 0
step = len(file_ids) // 1000
half = len(file_ids) // 2

print(datetime.now(), 'started calculating overall
        frequencies')

pool = ThreadPool(threads)
results = pool.map(get_text_freq, tqdm_notebook(file_ids
        ))
pool.close()

print(datetime.now(), 'ended calculating overall
        frequencies')

# Calculating normalized frequencies

for token_length in text_stats['total']:
    overall_freq = sum(text_stats['total'][token_length
        ].values())

    for file_id in text_stats['total'][token_length]:

```



```

absfreq = text_stats['total'][token_length][
    file_id]
relfreq = Decimal(absfreq) / Decimal(
    overall_freq)
text_stats['total'][token_length][file_id] = {'
    absfreq': absfreq,
    'relfreq': float(relfreq)}

text_stats_insert = token_stats.insert_one({'_text-stats
    ': True,
                                           'text_stats':
                                           text_stats})

text_stats = token_stats.find_one({'_text-stats': True})
    ['text_stats']

# # Part III: Calculate DP by Gries

# Calculate Stephan Th. Gries's (2008, 2010) "deviation
    of proportions", in order to see how well dispersed
    tokens are in your corpus

def calc_dp(token):

    global text_stats, token_stats

    try:
        # See if DP has already been calculated
        dp = token_stats.find_one({'token': token})['dp'
            ]
        return dp

    except TypeError:
        # Token not found in DB
        return float(1)

    except KeyError:
        # DP was not calculated yet
        # Let's calculate it then

        try:

```



```

    except TypeError:
        # Token not found in DB
        return float(1)

    except KeyError:
        return float(1)

# If this operation fails,
# try dividing the MongoDB request more.
# For instance:
# first you run this cell for tokens with more than
# 1,000 occurrences
# all_tokens = token_stats.distinct('token', {'freq': {'
#     $gt': 1000}})
# then you run this cell for tokens with less than 1,000
# occurrences
# all_tokens = token_stats.distinct('token', {'freq': {'
#     $lt': 1000}})
# It will all depend on the size of your data.

all_tokens = token_stats.distinct('token', {'dp': {'
    $exists': False}})
tokens_processed = 0

print(datetime.now(), 'adding DP values for', len(
    all_tokens), 'tokens')
pool = ThreadPool(threads)
results = pool.map(calc_dp, tqdm_notebook(all_tokens))
pool.close()
print(datetime.now(), 'finished adding DP values for',
    len(all_tokens), 'tokens')

```

B.3 THE TEXT DIFFICULTY ANALYZER + DIFFICULTY HIGHLIGHTER

Listing B.3: The Text Difficulty Analyzer + Difficulty Highlighter

```

from nltk import sent_tokenize, TreebankWordTokenizer,
    ngrams, WhitespaceTokenizer
from itertools import accumulate, tee, chain
from collections import Counter
import itertools

```

```
from datetime import datetime, date, time
from multiprocessing.dummy import Pool as ThreadPool
from collections import defaultdict, OrderedDict
import os, os.path
import re
import string
from decimal import Decimal
from decimal import *
getcontext().prec = 6
import statistics
import pymongo
from pymongo import MongoClient
from tqdm import tqdm_notebook

# # Overall Settings

tokenization_language = 'hebrew' # check the NLTK
    sentence tokenizers for available languages
non_latin_alphabet = True # if your language uses a non-
    Latin alphabet, change to True
files_dir_or_wiki = 'wiki' # change to 'list' if you
    have a list of filenames
                                # or to 'dir' if you have a
                                specific directory
token_database = 'tokens-hebrew'
exclude_numbers = True # True / False
exclude_numbers_for_ngrams = False # True / False
number_of_ngrams = 1 # choose up to x number of n-grams
    to extract (minimum = 1)
threads = 4 # decrease this number if you have an old/
    low core processor (minimum = 1)
chosen_encoding = 'utf-8-sig' # better utf-8-sig than
    utf-8, saves trouble

if number_of_ngrams <= 1:
    ngramrange = [1]
elif number_of_ngrams > 1:
    ngramrange = range(1, number_of_ngrams+1)

if non_latin_alphabet == False:
    tokenizer = TreebankWordTokenizer()

elif non_latin_alphabet == True:
```

```
try:
    from polyglot.downloader import downloader
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install polyglot')

try:
    import PyICU, icu
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install PyICU')

try:
    import pycld2
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install pycld2 ')
```

List of ICU-supported languages

```
langs = {'afrikaans': 'af', 'alemannic': 'als', 'amharic': 'am', 'aragonese': 'an', 'arabic': 'ar', 'egyptian arabic': 'arz', 'assamese': 'as', 'asturian': 'ast', 'azerbaijani': 'az', 'bashkir': 'ba', 'bavarian': 'bar', 'belarusian': 'be', 'bulgarian': 'bg', 'bangla': 'bn', 'tibetan': 'bo', 'bishnupriya': 'bpy', 'breton': 'br', 'bosnian': 'bs', 'catalan': 'ca', 'chechen': 'ce', 'cebuano': 'ceb', 'czech': 'cs', 'chuvash': 'cv', 'welsh': 'cy', 'danish': 'da', 'german': 'de', 'zazaki': 'diq', 'divehi': 'dv', 'greek': 'el', 'english': 'en', 'esperanto': 'eo', 'spanish': 'es', 'estonian': 'et', 'basque': 'eu', 'persian': 'fa', 'finnish': 'fi', 'faroese': 'fo', 'french': 'fr', 'western frisian': 'fy', 'irish': 'ga', 'gan chinese': 'gan', 'scottish gaelic': 'gd', 'galician': 'gl', 'gujarati': 'gu', 'manx': 'gv', 'hebrew': 'he', 'hindi': 'hi', 'fiji hindi': 'hif', 'croatian': 'hr', 'upper sorbian': 'hsb', 'haitian creole': 'ht', 'hungarian': 'hu', 'armenian': 'hy', 'interlingua': 'ia', 'indonesian': 'id', 'iloko': 'ilo', 'ido': 'io', 'icelandic': 'is', 'italian': 'it', 'japanese': 'ja', 'javanese': 'jv', 'georgian': 'ka', 'kazakh': 'kk', 'khmer': 'km', 'kannada': 'kn', 'korean': 'ko', 'kurdish': 'ku', 'kyrgyz': 'ky',
```

```

latin': 'la', 'luxembourgish': 'lb', 'limburgish': '
li', 'lombard': 'lmo', 'lithuanian': 'lt', 'latvian'
: 'lv', 'malagasy': 'mg', 'macedonian': 'mk', '
malayalam': 'ml', 'mongolian': 'mn', 'marathi': 'mr'
, 'malay': 'ms', 'maltese': 'mt', 'burmese': 'my', '
nepali': 'ne', 'dutch': 'nl', 'norwegian nynorsk': '
nn', 'norwegian': 'no', 'occitan': 'oc', 'odia': 'or
', 'ossetic': 'os', 'punjabi': 'pa', 'pampanga': '
pam', 'polish': 'pl', 'piedmontese': 'pms', 'pashto'
: 'ps', 'portuguese': 'pt', 'quechua': 'qu', '
romansh': 'rm', 'romanian': 'ro', 'russian': 'ru', '
sanskrit': 'sa', 'sakha': 'sah', 'sicilian': 'scn',
'scots': 'sco', 'northern sami': 'se', 'serbo-
croatian': 'sh', 'sinhala': 'si', 'slovak': 'sk', '
slovenian': 'sl', 'albanian': 'sq', 'serbian': 'sr',
'sundanese': 'su', 'swedish': 'sv', 'swahili': 'sw'
, 'silesian': 'szl', 'tamil': 'ta', 'telugu': 'te',
'tajik': 'tg', 'thai': 'th', 'turkmen': 'tk', '
tagalog': 'tl', 'turkish': 'tr', 'tatar': 'tt', '
uyghur': 'ug', 'ukrainian': 'uk', 'urdu': 'ur', '
uzbek': 'uz', 'venetian': 'vec', 'vietnamese': 'vi',
'west flemish': 'vls', 'volapük': 'vo', 'walloon':
'wa', 'waray': 'war', 'yiddish': 'yi', 'yoruba': 'yo
', 'chinese': 'zh', 'chinese character': 'zhc', '
chinese word': 'zhw'}

```

```

if tokenization_language.lower() in langs:

    token_models = []

    for lang in langs:
        code = langs[lang]

        if tokenization_language in lang:
            token_models.append(code)

    print(tokenization_language.capitalize(), 'supported
        with models:', token_models)

elif tokenization_language.lower() not in langs:
    print(tokenization_language.capitalize(), 'not
        supported. Reverting to Treebank')
    tokenizer = TreebankWordTokenizer()

```

```
# Downloading polyglot language models for the language
# you've chosen
# COMMENT THIS SECTION AFTER YOU DOWNLOAD THE MODELS

for model in token_models:

    downloader.download('LANG:'+model)

try:
    from polyglot.text import Text
except ModuleNotFoundError:
    get_ipython().system('{sys.executable} -m pip
        install morfessor')

from polyglot.text import Text

def icu_tokenizer(text_as_string):

    global token_models

    text = Text(text_as_string)
    text.language = token_models[0]

    return [str(word) for word in text.words]

def icu_sentence_tokenizer(text_as_string):

    global token_models

    text = Text(text_as_string)
    text.language = token_models[0]

    return [str(sentence) for sentence in text.sentences
        ]

# Token statistics
client = MongoClient()
db = client[token_database]

token_stats = db[token_database]
token_stats.allowDiskUse=True
```

```
text_stats = token_stats.find_one({'_text-stats': True})
    ['text_stats']
```

```
# # Prerequisite functions
```

```
def is_number_repl_isdigit(s):
    '''Returns True is string is a number,
    i.e. if it contains dot or comma.'''

    return re.sub('[.,]', '', s).isdigit()
```

```
def cleantokensfromsentence(tokens, bar_numbers):
    '''Clean a list of tokens to remove
    extraneous characters and numerals.
    Bar_numbers: if True, numbers will be deleted'''
```

```
clean_tokens = []
```

```
for token in tokens:
```

```
    if is_number_repl_isdigit(token) == True:
```

```
        if bar_numbers == True:
```

```
            '''Barring numbers'''
```

```
            pass
```

```
        elif bar_numbers == False:
```

```
            clean_tokens.append(token)
```

```
    elif token.isalnum() is False:
```

```
        if len(token) == 1:
```

```
            pass
```

```
        else:
```

```
            matches = 0
```

```
            for character in token:
```

```
                if character.isalnum() is False:
```

```
                    matches += 1
```

```
            if matches == len(token):
```

```
                pass
```

```
            else:
```

```
                clean_tokens.append(token)
```

```
    else:
```

```
        clean_tokens.append(token)
```

```
return clean_tokens
```



```
def getmultitokens(sentence, n_of_ngrams):
    '''From a list of tokens, generate up to n n-grams
    '''
    all_ngrams = []

    for x in range(2, n_of_ngrams+1):
        all_ngrams.extend(ngrams(sentence, x))

    return [' '.join(i) for i in list(chain(all_ngrams))
           ]

def opentextfile(text_filename):
    global chosen_encoding
    with open(text_filename, 'r', encoding=
              chosen_encoding) as t:
        return t.read()

def calc_dp(token):

    global text_stats, token_stats

    try:
        # See if DP has already been calculated
        dp = token_stats.find_one({'token': token})['dp'
        ]
        return dp

    except TypeError:
        # Token not found in DB
        return float(1)

    except KeyError:
        # DP was not calculated yet
        # Let's calculate it then

        try:
            document = token_stats.find_one({'token':
            token})
            token_length = document['len']
            token_freq = document['freq']
```

```

freq_in_files = dict(zip(document['
    occurred_in'],document['freq_occurred_in
    ']))

for file_id in freq_in_files:
    freq_in_files[file_id] = Decimal(
        freq_in_files[file_id]) / Decimal(
            token_freq)

differences = float(0)

for file_id in text_stats['total'][str(
    token_length)+'-grams']:

    expected_percentage = text_stats['total'
        ][str(token_length)+'-grams'][
        file_id]['relfreq']
    expected_percentage = abs(float(
        expected_percentage))
    observed_percentage = abs(float(
        freq_in_files.get(file_id, float(0))
        ))

    diff = abs(expected_percentage -
        observed_percentage)
    differences += diff

dp = Decimal(differences) / Decimal(2)

# Before returning DP, let us insert it to
# the DB

result = token_stats.update_one({'token':
    token},
                                {'$set': {'dp':
                                    float(dp)}})

return float(dp)

except TypeError:
    # Token not found in DB
    return float(1)

except KeyError:

```

```
        return(float(1))

# # The Text Difficulty Analyzer

def gather_tokens(text_as_string):

    global ngramrange, non_latin_alphabet

    gathered_tokens = {}

    for i in ngramrange:
        gathered_tokens[str(i)+'-grams'] = []

    # Convert to lowercase and into sentences

    textlc = text_as_string.lower()

    if non_latin_alphabet == False:
        sentences = sent_tokenize(textlc, language=
            tokenization_language)
    elif non_latin_alphabet == True:
        sentences = icu_sentence_tokenizer(textlc)

    for sentence in sentences:

        if non_latin_alphabet == False:
            original_tokens = tokenizer.tokenize(
                sentence)
        elif non_latin_alphabet == True:
            original_tokens = icu_tokenizer(sentence)

        clean_tokens = cleantokensfromsentence(
            original_tokens, exclude_numbers)
        gathered_tokens['1-grams'].extend(clean_tokens)

    if ngramrange != [1]:

        mtes = getmultitokens(
            cleantokensfromsentence(original_tokens,
                exclude_numbers_for_ngrams),
            number_of_ngrams)

        for mte in mtes:
```

```

        gathered_tokens[str(len(mte.split(' ')))
            + '-grams'].append(mte)

# Turn tokens into Counter dicts

for key in gathered_tokens:
    gathered_tokens[key] = dict(Counter(
        gathered_tokens[key]))

return gathered_tokens

def get_dp(counter):

    token = list(counter.keys())[0]

    dp = calc_dp(token)
    textfreq = list(counter.values())[0]

    for x in counter:

        return {'token': token,
            'dp': dp,
            'textfreq': textfreq}

def retrieve_dp_values(gathered_tokens):

    global threads

    for ngram_len in gathered_tokens:

        counter_dict = gathered_tokens[ngram_len]

        counter_list = [{x:counter_dict[x]} for x in
            counter_dict]

        pool = ThreadPool(threads)
        results = tqdm_notebook(pool.map(get_dp,
            counter_list))
        pool.close()

    for result in results:
        gathered_tokens[ngram_len][result['token']]
            = {'textfreq': result['textfreq'],

```

```
                'dp': result
                ['dp']]

    return gathered_tokens

def calc_tds(text_as_string):

    retrieved_tokens = retrieve_dp_values(gather_tokens(
        text_as_string))

    for ngram_len in retrieved_tokens:

        token_data = retrieved_tokens[ngram_len]

        dp_values = []
        unique_dp_values = []

        for token in token_data:

            textfreq = token_data[token]['textfreq']

            try:
                dp_value = float(token_data[token]['dp'
                    ])

                if dp_value != None:
                    unique_dp_values.append(dp_value)

                for f in range(textfreq):
                    if dp_value != None:
                        # Disregard tokens that weren't
                        # found in the DB
                        dp_values.append(dp_value)
            except TypeError:
                # Token does not exist on DB
                pass

        if len(dp_values) != 0 and len(dp_values) >= 2:
            retrieved_tokens[ngram_len]['_stats_'] =
                {'total_dp_values':
                 {'median': statistics.median(
                     dp_values),
                  'mean': statistics.mean(dp_values
                     )}},
                'unique_dp_values':
```

```

        {'median': statistics.median(
            unique_dp_values),
         'mean': statistics.mean(
            unique_dp_values),)}

    elif len(dp_values) == 0 or len(dp_values) < 2:
        print(datetime.now(), 'statistics not
            computed')

# The calculation of the TDS value below only
# considers 1-grams
# since they're much more important than longer n-
# grams
# but you can adapt it to include 2-grams and 3-
# grams

median_total = retrieved_tokens['1-grams']['_stats_'
    ]['total_dp_values']['median']
median_unique = retrieved_tokens['1-grams']['_stats_'
    ]['unique_dp_values']['median']

retrieved_tokens['tds'] = statistics.mean([
    median_total, median_unique])

return retrieved_tokens

# ## Difficulty Highlighter (HTML)

# Export as HTML a difficulty-highlighted version of the
# text.<br>
# It exports the HTML to the current working directory.

def difficulty_highlighter(text_filename):

    parameter = 'markdown_colors'

    text_as_string = opentextfile(text_filename)

    text_as_string = ' '.join(icu_tokenizer(
        text_as_string))

    text_as_string = re.sub('""[ ]', '"', text_as_string)
    text_as_string = re.sub('\'[ ]', "'", text_as_string)

```

```
text_as_string = re.sub('\n', "\n\n", text_as_string
)

print(datetime.now(), 'text file opened')

retrieved_tokens = calc_tds(text_as_string)

print(datetime.now(), 'tokens retrieved')

tds = retrieved_tokens.pop('tds', None)
stats = retrieved_tokens['1-grams'].pop('_stats_',
None)

difficulty_ranges = OrderedDict({'very easy': [0,
0.7],
'easy': [0.7, 0.80],
'average': [0.80, 0.90],
'slightly difficult': [0.90, 0.95],
'difficult': [0.95, 0.99],
'very difficult': [0.99, 1.00]})

parameters = OrderedDict({'html_colors': {'very easy
': 'color:LightGrey;',
'easy': 'color:green;',
'average': 'color:blue;',
'difficult': 'color:black;',
'slightly difficult': 'color:black;',
'very difficult': 'color:Crimson;'},

'markdown': {'very easy': 'font-style: normal;',
'easy': 'font-style: normal;',
'average': 'font-style: italic;',
'slightly difficult': 'color:black;',
'difficult': 'font-weight: bold;',
'very difficult': 'font-variant: small-caps;
letter-spacing: 1.5px'},

'markdown_colors': {'very easy': 'font-style
: normal;',
'easy': 'font-style: italic;',
'average': 'font-weight: 600;',
'slightly difficult': 'font-weight: bold;
color:navy;',
```

```

        'difficult': 'font-variant: small-caps;
                    letter-spacing: 1.5px',
        'very difficult': 'font-variant: small-
                          caps;font-weight:bold;letter-spacing:
                          1.5px;color:Crimson'}}))

if non_latin_alphabet == True:

    for key in parameters:

        if 'italic' in parameters[key]['easy']:

            parameters[key]['easy'] = 'text-
                                        decoration: underline;'

        if 'small-caps' in parameters[key]['
            difficult']:

            parameters[key]['difficult'] = '
                                            background-color: #DEDEDE;
                                            background-image: linear-gradient(to
                                                left, #F4F4F4 0%, #DEDEDE 100%);'

    for ngram_len in retrieved_tokens:

        for token in retrieved_tokens[ngram_len]:

            retrieved_tokens[ngram_len][token] =
                retrieved_tokens[ngram_len][token]['dp']

    token_ranges = {x:[] for x in difficulty_ranges}

    difficulties = []
    tokens = []
    dp_values = []

    for token in retrieved_tokens['1-grams']:

        token_dp = retrieved_tokens['1-grams'][token]

        for difficulty in difficulty_ranges:

            diffrange = difficulty_ranges[difficulty]

```



```

        if token_dp > diffrange[0] and token_dp <=
            diffrange[1]:

            difficulties.append(difficulty)
            token_modified = "(" + r"\b" + token + r
                "\b" + ")"
            tokens.append(token_modified)
            dp_values.append(token_dp)
            token_ranges[difficulty].append(token)

tokens_found = '<hr><h3>Wordlist:</h3><ol>'
for token_range in token_ranges:
    tokens_found+= ".join(["<li>",
        "<font style='font-size:small;%s'" % parameters[
            parameter][token_range],
        token_range.capitalize(),
        "]:</font>",
        " <font style='font-size:small;font-style:italic
            ;'>",
        ' | '.join(sorted(token_ranges[token_range])),
        '</font></li>',
    ])
tokens_found += '</ol>'

dp_values = [str(x)[:5] for x in dp_values]

colors = [parameters[parameter][x] for x in
    difficulties]

compiler = '|'.join(tokens)
regex = re.compile(compiler, re.I)

i = 0
output = ""<!DOCTYPE html>
<html>
<head>
<link href="https://fonts.googleapis.com/css?family=
    Vollkorn:400,400i,600,600i,700,700i,900,900i&
    ;subset=latin-ext" rel="stylesheet">
<style>
body {
font-family: 'Vollkorn', 'Georgia', serif;
}
</style>

```

```

</head>
<body>
<h1>Text Difficulty Analyzer v. 1.00</h1>
<h3><i>Difficulty Highlighter</i></h3>
Filename: ""

output = re.sub('\n', '', output)

output += text_filename + '<br>Text Difficulty Value
      : <b>' + str(tds)[:8] + '</b><hr>'

legend = '<b>LEGEND:</b> '

for x in parameters[parameter]:

    legend += "".join(["<font style='",
                      parameters[parameter][x],
                      "'>",
                      x.capitalize(),
                      '</font> | '])

output += legend + "<hr><h3><font style='font-
      variant:small-caps;letter-spacing: 1.5px'>
      Highlighted Text:</font></h3>"

print(datetime.now(), 'initiating difficulty
      highlighter')

for m in regex.finditer(text_as_string):

    output += "".join([text_as_string[i:m.start()],
                      "<font style='%s'" %
                      colors[m.lastindex
                      -1],
                      text_as_string[m.start():
                      m.end()],
                      # Uncomment line below if you
                      # wish to have DP values
                      # alongside

```

```

                "</font><font style='
                    vertical-align:super;
                    font-size:8pt;color:
                    LightGrey'>%s" %
                    dp_values[m.lastindex
                    -1],
                "</font>"] )
        i = m.end()

html = "".join([output, text_as_string[m.end():],
                tokens_found, "</body></html>"])

html = re.sub('\n', '<br>', html)

output_filename = text_filename.split('\')
                [-1][:-4]+'_highlighted.html'

with open(output_filename, 'w', encoding=
          chosen_encoding) as o:
    o.write(html)

print(datetime.now(), 'exported as file',
        output_filename, ' -- Process complete')

#return token_ranges

#return difficulties, tokens, compiler, regex, html

def difficulty_highlighter_latex(text_filename):

    parameter = 'latex'

    text_as_string = opentextfile(text_filename)

    text_as_string = ' '.join(icu_tokenizer(
        text_as_string))

    text_as_string = re.sub('""', '"', text_as_string)
    text_as_string = re.sub('\'', "'", text_as_string)
    text_as_string = re.sub('\n', "\n\n", text_as_string
        )

    print(datetime.now(), 'text file opened')
```

```

retrieved_tokens = calc_tds(text_as_string)

print(datetime.now(), 'tokens retrieved')

tds = retrieved_tokens.pop('tds', None)
stats = retrieved_tokens['1-grams'].pop('_stats_',
    None)

difficulty_ranges = OrderedDict({'very easy': [0,
    0.7],
    'easy': [0.7,
    0.80],
    'average': [0.80,
    0.90],
    'slightly difficult': [0.90,
    0.95],
    'difficult': [0.95,
    0.99],
    'very difficult': [0.99, 1.00]})

parameters = OrderedDict({'latex':
    {'very easy': '\\veryeasy{
    ',
    'easy': '\\emph{',
    'average': '\\average{',
    'slightly difficult': '\\
    slightlydifficult{',
    'difficult': '\\difficult
    {',
    'very difficult': '\\
    verydifficult{'},
    'latex_black':
    {'very easy': '\\textrm{',
    'easy': '\\emph{',
    'average': '\\underline{
    ',
    'slightly difficult': '\\
    slightlydifficult{',
    'difficult': '\\difficult
    {',

```

```

        'very difficult': '\\
            verydifficult{',
        #'font-variant: small-
            caps;font-weight:bold
            ;letter-spacing: 1.5
            px;color:Crimson'
    })

if non_latin_alphabet == True:

    for key in parameters:

        if 'emph' in parameters[key]['easy']:

            parameters[key]['easy'] = '\\underline{

for ngram_len in retrieved_tokens:

    for token in retrieved_tokens[ngram_len]:

        retrieved_tokens[ngram_len][token] =
            retrieved_tokens[ngram_len][token]['dp']

token_ranges = {x:[] for x in difficulty_ranges}

difficulties = []
tokens = []
dp_values = []

for token in retrieved_tokens['1-grams']:

    token_dp = retrieved_tokens['1-grams'][token]

    for difficulty in difficulty_ranges:

        diffrange = difficulty_ranges[difficulty]

        if token_dp > diffrange[0] and token_dp <=
            diffrange[1]:

            difficulties.append(difficulty)
            token_modified = "(" + r"\b" + token + r
                "\b" + ")"

```

```

        tokens.append(token_modified)
        dp_values.append(token_dp)
        token_ranges[difficulty].append(token)

tokens_found = '\n\n\\section{Wordlist}\n\n\\begin{
itemize}'
for token_range in token_ranges:
    tokens_found+= ".join(["\n\\item ",
                           "%s" % parameters[
                               parameter][
                                   token_range],
                           token_range.capitalize(),
                           "}: ",
                           "{\\emph{" ,
                           ' | '.join(sorted(
                               token_ranges[
                                   token_range])),
                           '}}}',
                           ])
tokens_found += '\\end{itemize}\n'

dp_values = [str(x)[:5] for x in dp_values]

colors = [parameters[parameter][x] for x in
difficulties]

compiler = '|'.join(tokens)
regex = re.compile(compiler, re.I)

i = 0
output = r""""% !TEX TS-program = xelatex
% !TEX encoding = UTF-8

% This is a simple template for a XeLaTeX document
using the "article" class,
% with the fontspec package to easily select fonts.

\documentclass[11pt]{article} % use larger type;
default would be 10pt

\usepackage{fontspec} % Font selection for XeLaTeX;
see fontspec.pdf for documentation
\defaultfontfeatures{Mapping=tex-text} % to support
TeX conventions like ``---''

```

```

\usepackage{xunicode} % Unicode support for LaTeX
    character names (accents, European chars, etc)
\usepackage{xltextra} % Extra customizations for
    XeLaTeX
\usepackage[usenames, dvipsnames]{color}""

if parameter == 'latex_black':
    output += ""
    \definecolor{crimson}{RGB}{0,0,0}
    \definecolor{navy}{RGB}{0,0,0}
    \definecolor{lightgray}{RGB}{192,192,192}
    ""

else:
    output += ""
    \definecolor{crimson}{RGB}{255,0,64}
    \definecolor{navy}{RGB}{64,0,255}
    \definecolor{lightgray}{RGB}{192,192,192}
    ""

#output = re.sub('\n', '', output)

if '/' in text_filename:
    filename = text_filename.split('/')[-1]
elif '\\' in text_filename:
    filename = text_filename.split('\\')[-1]

filename = re.sub("_", '\_', filename)

output += r""

\setmainfont{GaramondPremrPro-Med}[
Extension = .otf,
BoldFont={GaramondPremrPro-Bd},
ItalicFont={GaramondPremrPro-MedIt},
BoldItalicFont={GaramondPremrPro-BdIt},
Numbers=OldStyle,
]

\newfontfamily\semibold{GaramondPremrPro-Smbd}

\newfontfamily\lightfont{GaramondPremrPro}

```

```

\newcommand{\verydifficult}[1] {\begingroup\textsc{\
    textbf{\textcolor{crimson}{#1}}}\endgroup}

\newcommand{\difficult}[1] {\begingroup
\lightfont{\textsc{#1}}\endgroup}

\newcommand{\slightlydifficult}[1] {\begingroup \
    textcolor{navy}{\textbf{#1}}\endgroup}

\newcommand{\average}[1]{\begingroup \semibold{#1}\
    endgroup}

\newcommand{\veryeasy}[1]{#1}

\newcommand{\dpsuper}[1] {
\lightfont{\textcolor{lightgray}{\textsuperscript
    {#1}}}}
}

% other LaTeX packages.....
\usepackage{geometry} % See geometry.pdf to learn
    the layout options. There are lots.
\geometry{a4paper} % or letterpaper (US) or a5paper
    or....
%\usepackage[parfill]{parskip} % Activate to begin
    paragraphs with an empty line rather than an
    indent

\usepackage{graphicx} % support the \includegraphics
    command and options

\title{""+filename+r""}
\author{Difficulty Highlighter v. 1.00}
%\date{} % Activate to display a given date or no
    date (if empty),
    % otherwise the current date is printed

\begin{document}
\maketitle
""

output += filename + '\n\nText Difficulty Scale: \
    textbf{' + str(tds)[:8] + '}\n'

```



```

legend = '\n\\textbf{LEGEND ---} '

for x in parameters[parameter]:

    legend += "".join(["",
                        parameters[parameter][x],
                        "",
                        x.capitalize(),
                        '} | '])

output += "\n\\section{Highlighted Text}" + '\\
bigskip \n\n' + legend + '\\bigskip \n\n'

print(datetime.now(), 'initiating difficulty
highlighter')

for m in regex.finditer(text_as_string):

    output += "".join([text_as_string[i:m.start()],
                        "%s" % colors[m.lastindex
                        -1],
                        text_as_string[m.start():
                        m.end()],
                        # Uncomment line below if you
                        wish to have DP values
                        alongside
                        "}\dpsuper{%s" %
                        dp_values[m.lastindex
                        -1],
                        "]" ] )

    i = m.end()

html = "".join([output, text_as_string[m.end():],
tokens_found, "\\end{document}"])

output_filename = text_filename.split('\\')[ -1 ][ :-4 ]
+ '_highlighted.tex'

with open(output_filename, 'w', encoding=
chosen_encoding) as o:
    o.write(html)

```

```
print(datetime.now(), 'writing to file',  
      output_filename, ' -- Process complete')  
  
difficulty_highlighter('./bach_he.txt')  
  
difficulty_highlighter_latex('./bach_he.txt')
```



ASSEMBLING THE ACADEMIC AND SPOKEN CORPORA

C.1 OBTAINING TDVS FROM THE SKETCHENGINE WORDLISTS

The short script shown in listing C.1 was used to obtain values of **DP** and **TDV** from the SketchEngine wordlists for both the English and Portuguese experiments.

Listing C.1: Obtaining **DP** and TDVs from SketchEngine wordlists

```
total_occurrences = []
unique_occurrences = []
not_found = []
not_found_total = []
freq_dict = {}
with open('sketch-engine-wordlist.txt', 'r', encoding=
    chosen_encoding) as t:
    t = t.read()
    t = t.split('\n')
    for token in t:
        tab_split = token.split('\t')
        token = tab_split[0]
        freq = int(tab_split[1])
        dp = calc_dp(token)
        freq_dict[token] = {'freq': freq,
                            'dp': dp}
        if dp != None:
            total_occurrences.extend([dp] * freq)
            unique_occurrences.append(dp)

    elif dp == None:
        # Comment the next two lines if
        # one wishes to disregard tokens that were
        # not found
        total_occurrences.extend([1.0] * freq)
        unique_occurrences.append(1.0)
```

```

        not_found.append(1.0)
        not_found_total.extend([1.0] * freq)

print('Median of unique DP values', statistics.median(
    unique_occurrences))
print('Mean of unique DP values', statistics.mean(
    unique_occurrences))
print('Median of total DP values', statistics.median(
    total_occurrences))
print('Mean of total DP values', statistics.mean(
    total_occurrences))
print('Unique tokens not found', len(not_found))
print('Total tokens not found', len(not_found_total))
print('TDV', statistics.mean([statistics.median(
    unique_occurrences), statistics.median(
    total_occurrences)]))

```

C.2 ENGLISH

C.2.1 *The academic text corpus*

C.2.1.1 *Choosing academic articles randomly*

Listing C.2: Choosing academic article files randomly

```

import glob
import random
import shutil

for letter in ['P', 'H', 'L', 'S']:
    # Each letter means one article subject area. We
    # need 10 of each

    academic = random.sample(glob.glob(inputdir+letter+'
        *.txt'), 10)

    for article in academic:
        if article not in glob.glob(outputdir+'*.txt'):
            shutil.copy(article, outputdir)

```

C.2.2 *The spoken text corpus*

We worked with the .trn version of the transcription files of Santa Barbara, since they seemed a little more amenable to work with than the .chat files. With all 60 .trn files extracted, on a directory, we needed to clean them to remove transcription signs and words.

Listing C.3: Cleaning the Santa Barbara corpus files

```
import glob
import re

trn = glob.glob(sbtranscriptionfolder+'*.trn')
n = 0

for trn_file in trn:

    with open(trn_file, 'r') as f:
        chat = f.read()

    clean_transcription = ''

    for line in chat.split('\n'):

        dialogue = line.split('\t')[-1]

        to_delete = ""<HI|HI>|M\[hm\]|u" "um,|u" "um|Uh
|unh|Unh|uhu" " |Unhm|hmm|HmM|Mmm|Mhm|Mn|hm|
Hm|Mm|Mhm|COUGH|\(HI\)|\(HII\)|\(Hx\)|\(H\)|
<I|Hx|I>|<P|P>|<W|W>|<L|L>|AR|<SM|SM>|<PAR|
PAR>|<SING|SING>|<SNAP|SNAP>|<VOX|VOX>|<SNA|
WH|SNA>|<YWN|YWN>|<BK|BK>|X*|_|<@|@>|TSK|<F|
F>|<X|X>|<Q|Q>|\.|TROAT|THROAT|CLAP|SWALLOW|
MICROPHONE|SLAPPING|SM|SNIFF|CLICK|FOOD|OOD|
SNEEZE|KISS|LATERAL|GROWL|SIGH|WHISTLE|ISTLE
|HISS|MRC|THUM|HOWL|YAWN|[\\(\\)]|GASP|GROAN|
SLAP|[0123456789_\\%~\\[\\]*@&<!]|SCREAM|
SNORT|[^0-9]\\-|HISD|CHOKE|HL|BK|LAUGHTER|MIC
|\\+|""".split('|')

    for char in to_delete:
        dialogue = re.sub(char, '', dialogue)
    dialogue = re.sub(' ', ' ', dialogue)
```

Table C.1: **SBCSAE** transcription file before and after cleaning.

Original transcription file	Clean transcription file
... So you don't need to go ... borrow equipment from anybody, to –	So you don't need to go borrow equipment from anybody, to-
... to do the feet?	to do the feet?
... [Do the hooves]?	Do the hooves?
[(H)=] <YWN Well, we're gonna have to find somewhere, to get, (Hx) ... something (Hx) YWN>.	Well, we're gonna have to find somewhere, to get, something
.. So, [~Mae-] –	So, Mae-
[I'm gonna] (Hx) –	I'm gonna -
[2~Mae ~Lynne XX2]	Mae Lynne
[2(H) We're not2] gonna do the feet today, I'm gonna wait till like, early in the morning=, .. to do those, cause y- –	We're not gonna do the feet today, I'm gonna wait till like, early in the morning, to do those, cause -
I mean you get s=o ti=red. (H) ... n- you just, ... it takes % –	I mean you get so tired you just, it takes -

```

if '$' in dialogue:
    # $ indicates commentary
    pass
else:
    clean_transcription += dialogue+'\n'

n += 1

filename = 'SBC'+ '{:03}'.format(n) + '.txt'

with open(outputfolder+filename, 'w',
          encoding=chosen_encoding) as f:
    f.write(clean_transcription)

```

Table C.1 shows the before and after of the corpus cleaning.

C.3 EXAMPLE OF THE CLEANING RESULTS FOR PORTUGUESE

Table C.2 shows the before and after of the corpus cleaning for the COB.

Table C.2: COB transcription file before and after cleaning.

Original transcription file	Clean transcription file
Cod Utt. Spe. File	
bfamcvo1 1 LEO o Juninho <foi> //	o Juninho foi
bfamcvo1 2 GIL <ô / mas> / voltando à questão / falando em e também falando em povo mascarado / esse povo do Galáticos é muito palha / eu acho que es nũ deviam mais participar / e <tal> //	ô mas voltando à questão falando em e também falando em povo mascarado esse povo do Galáticos é muito palha eu acho que es nũ deviam mais participar e tal
bfamcvo1 3 LUI <não> //	não
bfamcvo1 4 LEO <não> //	não
bfamcvo1 5 LUI <eu acho não> //	eu acho não
bfamcvo1 6 LEO <com certeza> //	com certeza
bfamcvo1 7 LUI <com certeza es nũ vão participar / uai> //	com certeza es nũ vão participar uai
bfamcvo1 8 LEO <eles são piores do que o> Durepox //	eles são piores do que o Durepox
bfamcvo1 9 EVN é / pois <é> //	é pois é
bfamcvo1 10 LUI <agora> manda uma barrinha <minha> //	agora manda uma barrinha minha
bfamcvo1 11 EVN <porque o Durepox> / pelo menos jogava bola //	porque o Durepox pelo menos jogava bola
bfamcvo1 12 GIL não / e o Durepox / eu vou +	não e o Durepox eu vou
bfamcvo1 13 GIL tinha um cara //	tinha um cara
bfamcvo1 14 GIL era &aque [/2] era &aque [/2] era aquele cara <lá / que era muito> / <muito> / muito <palha> //	era aque era aque era aquele cara lá que era muito muito muito palha
bfamcvo1 15 EVN <era aquele cara> //	era aquele cara
bfamcvo1 16 EVN <é> //	é
bfamcvo1 17 LUI <escroto> / <e como> ele era amigo dos caras / a galera meio que tomava / as dores //	escroto e como ele era amigo dos caras a galera meio que tomava as dores
bfamcvo1 18 LUI mas nũ [/2] mas nũ [/2] eles nũ eram todos <escrotos> / igual o pessoal do Galáticos não //	mas nũ mas nũ eles nũ eram todos escrotos igual o pessoal do Galáticos não
bfamcvo1 19 GIL <é> //	é

D

DIFFICULTY HIGHLIGHTER TRANSLATIONS

We performed automated translations (through Google Translate¹) of the difficulty-highlighted .html files into English; converted them to .pdf, and then to .png using Adobe Photoshop. The translations are likely not very accurate, and there are inconsistencies in the formatting of the translations (especially in the case of Hebrew and Japanese), but they provide a cursory view of the difficulty estimation process for individual tokens in the texts we used. The images show, again, only part of the automated translations.

¹ <https://translate.google.com/>. Accessed on 09/30/2018.

JOHANN^{0.982} SEBASTIAN^{0.986} **BACH**^{0.992} (**EISENACH**^{0.999}, 21^{0.139}
 of^{0.139} *March*^{0.713} of^{0.139} 1685 - **LEIPZIG**^{0.995}, 28 of^{0.139} *July*^{0.709}
 of^{0.139} 1750) was^{0.329} a^{0.239} **composer**^{0.932},
HARPSICHORDIST^{0.999}, **KAPPELLMEISTER**^{0.999}, CONDUCTOR^{0.961}
 , **ORGANIST**^{0.998}, teacher^{0.855}, **VIOLINIST**^{0.996} and^{0.142}
VIOLIST^{1.0} **ORIGINATING**^{0.990} from^{0.218} **HOLY**^{0.985} **Empire**^{0.866}
ROMAN-GERMANIC^{0.866}, *current*^{0.752} **Germany**^{0.866}.
 Born^{0.898} *in a*^{0.702} *family*^{0.740} of^{0.139} long^{0.868} tradition^{0.861}
 musical^{0.869}, **early**^{0.936} **showed**^{0.900} **has**^{0.917} **TALENT**^{0.959}
 and^{0.142} *logo*^{0.715} *became*^{0.736} a^{0.239} **musician**^{0.943} full^{0.883}.
STUDENT^{0.952} **INDEFATIGABLE**^{0.995}, **acquired**^{0.947} one^{0.239}
WIDE^{0.965} **knowledge**^{0.861} of^{0.219} **Music**^{0.802} **European**^{0.893} of^{0.139}
 his^{0.372} time^{0.659} and^{0.142} of^{0.377} **generations**^{0.925}
previous^{0.834}. He **PLAYED**^{0.968} many^{0.614} **positions**^{0.948} to^{0.187}
CUTS^{0.953} and^{0.142} **churches**^{0.944} **GERMAN**^{0.977}, but^{0.434}
 their^{0.496} **functions**^{0.897} more^{0.324} **HIGHLIGHTED**^{0.992} were^{0.490}
 the^{0.157} of^{0.139} **KANTOR**^{1.0} of the^{0.219} **church**^{0.848} to^{0.139}
 Saint^{0.487} **THOMAS**^{0.973} and^{0.142} **Director**^{0.850} **Musical**^{0.869}
 the^{0.219} city^{0.681} of^{0.139} **LEIPZIG**^{0.995}, where^{0.497} **developed**
 the^{0.908} part^{0.157} end^{0.509} and^{0.634} over^{0.142} *important*^{0.324} ^{0.714}

Figure D.1: *Johann Sebastian Bach* Portuguese WP article translated into English.

Yo Han , **SEBASTIAN** - Bach (**JOHANN**
SEBASTIAN BACH , 1685 year 3 March 31
 days (**JULIAN** calendar 1685 year 3 May
 21 days) - 1750 years 7 May 28 Japan)
 is , 18 century of Germany at active
 was composer - musician at a .
BAROQUE music of important a
 composer of one person at , **KEYBOARD**
 instruments of **MUSICIANS** as also
 high people at Yes , then from
 improvisation **performance** of **LANDLORD**
 as intellectual et Les Tei was .
 Bach **researchers** of **views** at , Bach
 The **BAROQUE** music of last **Tails**
 to position to composer as It
 to of the music to culmination and
 both evaluation of is is , posterity
 To is , western music of **basis**
 to **building** was composer at Yes
 music of headwaters at a both

Figure D.2: *Johann Sebastian Bach* Japanese WP article translated into English.

'Johan ^{0.960} **SEBASTIAN** ^{0.999} **BACH** ^{1.0} (to ^{0.113} German ^{0.776} :
JOHANN ^{0.999} **SEBASTIAN** ^{1.0} **BACH** ^{1.0}) (born 21 March ^{0.805}
1685 - died July 28, 1750) **Composer** ^{0.961} and ^{0.114} musician
Citadel ^{0.878} and ^{0.114} **HARPSICHORD** ^{1.0} from ^{0.780} **Germany**
was ^{0.831}, which ^{0.342} works is ^{0.965} for ^{0.275} **Category**
singing ^{0.942}, **ORCHESTRA** ^{0.994} and ^{0.114} **single** ^{0.839}
instruments ^{0.944}, almost all ^{0.509} of ^{0.729} different from
Music ^{0.762} of **BAROQUE** ^{0.995} to ^{0.222} **CONTAIN** ^{1.0} to be
and ^{0.734} music ^{0.114} to ^{0.797} times ^{0.223} to ^{0.470} to ^{0.222} to ^{0.113}
maturity of ^{0.933} of its ^{0.726} **has** ^{0.984} Is ^{0.274}. With ^{0.169} of ^{0.409}
than ^{0.493} He ^{0.527} of ^{0.545} just ^{0.710} of ^{0.683} to ^{0.222} from ^{0.105}
Music ^{0.797} provider of ^{0.592} was ^{0.773}, but ^{0.456} a ^{0.633} style
prevalent ^{0.974} to ^{0.095} music ^{0.797} **Germany** ^{0.831} to ^{0.222} with
Technology ^{0.169} **COUNTERPOINT** ^{0.971}, **Rich** ^{1.0}, ^{0.895}, ^{0.871}. The
Technology ^{0.168}, about ^{0.971} to ^{0.700} set ^{0.236} **SORT** ^{0.827} ^{0.997}
HARMONIC ^{0.999} and ^{0.114} **MOTIF** ^{0.999} on a ^{0.095} **scale** ^{0.927}
Small ^{0.599} and ^{0.114} **Large** ^{0.423} respectively ^{0.342} and ^{0.114} as
well as ^{0.346} **SUITABILITY** ^{0.998} **RHYTHM** ^{0.998} **and** ^{0.919}, and
Texture ^{0.114} to ^{0.802} from ^{0.940} from ^{0.736} Music ^{0.105} of ^{0.797}
Other ^{0.604} since ^{0.286} **Italy** ^{0.483} and ^{0.839} **France** ^{0.114} to ^{0.688}

Figure D.3: *Johann Sebastian Bach* Persian WP article translated into English.

JOHAN SEBASTIEN BACH (German : JOHANN
 SEBASTIAN BACH (Information Help
); 21 March 1685-28 July, in 1750) was composer
 German period BAROQUE , BELONGING
 TO with a PLAYERS ORGAN AND COMPOSERS
 biggest most each times . His works
 of BACH , ADDRESSES with EXTRACTION
 Art AND TECHNICAL of EQUIPMENT
 COUNTERPOINT , UNCHARTED THE DEPTHS
 INTELLECTUAL AND BEAUTY artistic and Yes
 Many seen BACH to composer
 brought a peak to music POLYPHONIC
 . His time learned largely PERFORMER
 ORGANIST VIRTUOSO , and death WANED
 FAME , but She SEATING AWARENESS
 Public in The - 19 and PESA since .
 Between works most important AND
 SELLERS best : TOCCATA AND FUGUE D
 MINOR , PIANO EQUATED , AND VARIATIONS
 Goldberg , MATTHEWS PHEASANT ,
 JOHANNES PASSION , CONCERTI

Figure D.4: *Johann Sebastian Bach* Hebrew WP article translated into English.

TOKEN LISTS

This chapter includes lists of the top 100 1-grams listed by descending absolute frequency, together with their DP values and dispersion values (defined as the number of articles the token appeared in). The # columns denote the ranks, so one can see how different the frequency, DP, and dispersion rankings are.

For the Portuguese, Hebrew, Japanese, and Persian lists, we translated the tokens using Google Translate¹ into English.

E.1 ENGLISH

Table E.1: Top 100 English 1-grams

Token	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
the	4681321	1	0.109	1	113055	1
of	2232401	2	0.154	6	105796	4
and	1870987	3	0.113	2	98693	5
in	1816669	4	0.144	4	108883	2
to	1356457	5	0.150	5	92471	7
a	1342505	6	0.142	3	107914	3
was	764512	7	0.276	14	84186	8
is	635055	8	0.356	23	98024	6
as	547143	9	0.200	7	75132	9
for	541226	10	0.221	10	74820	10
on	504288	11	0.267	13	73571	12
with	468618	12	0.212	8	71012	13
by	463092	13	0.215	9	74468	11
that	394762	14	0.302	16	55009	19
's	389686	15	0.340	20	59438	18

¹ <https://translate.google.com/>. Accessed on 09/23/2018.

Table E.1 continued from previous page

Token	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
from	340770	16	0.243	11	69100	14
at	328806	17	0.305	17	67344	15
his	312200	18	0.545	84	41968	28
it	304046	19	0.324	19	65774	16
an	251638	20	0.256	12	65314	17
were	231626	21	0.417	38	42463	27
are	224030	22	0.481	57	42577	26
which	210972	23	0.281	15	53008	20
or	182965	24	0.473	51	39460	33
be	181152	25	0.383	26	40125	31
this	180416	26	0.350	21	45507	23
also	166349	27	0.317	18	52747	21
had	161828	28	0.422	40	38748	35
has	153677	29	0.415	36	46609	22
first	140088	30	0.393	28	44513	24
their	135859	31	0.420	39	35148	40
not	134305	32	0.385	27	35383	39
one	130848	33	0.351	22	43624	25
its	130738	34	0.432	43	40195	30
but	127409	35	0.363	24	38592	36
after	122789	36	0.397	29	40852	29
have	120872	37	0.412	34	34173	42
new	118115	38	0.485	59	34869	41
they	117079	39	0.447	47	31874	47
who	117042	40	0.427	41	40106	32
been	103416	41	0.398	30	36050	37
two	103381	42	0.401	31	38923	34
other	99645	43	0.379	25	35789	38
when	91415	44	0.405	32	33538	43
there	87595	45	0.477	54	30343	49
all	86668	46	0.411	33	32576	45

Table E.1 continued from previous page

Token	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
during	84497	47	0.450	48	32150	46
into	82363	48	0.413	35	31615	48
more	80686	49	0.417	37	29698	51
time	77630	50	0.438	44	30307	50
may	73473	51	0.550	87	32801	44
most	72232	52	0.442	45	28540	56
years	69861	53	0.502	68	29418	52
some	69527	54	0.466	50	26123	63
only	69449	55	0.432	42	28766	55
over	69002	56	0.447	46	29028	53
such	68913	57	0.505	70	24163	72
would	68543	58	0.512	75	21710	90
used	65841	59	0.571	95	22783	84
between	64738	60	0.485	60	27155	59
many	61152	61	0.480	56	25078	68
where	59983	62	0.486	61	28425	57
later	59748	63	0.494	64	27061	60
up	58844	64	0.474	53	27457	58
these	58539	65	0.509	74	22087	86
than	58394	66	0.478	55	23347	76
about	57772	67	0.509	73	25373	67
while	57269	68	0.462	49	24989	69
made	56092	69	0.494	63	26493	61
out	56059	70	0.497	66	24391	71
under	56037	71	0.540	83	24816	70
then	55156	72	0.503	69	25765	66
known	53990	73	0.526	78	28837	54
three	53974	74	0.509	72	25805	65
no	52121	75	0.508	71	23031	80
became	51361	76	0.550	88	23920	74
year	51079	77	0.584	99	23065	79

Table E.1 continued from previous page

Token	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
part	51047	78	0.527	80	26307	62
including	49437	79	0.496	65	25902	64
through	49168	80	0.513	76	23025	81
both	49150	81	0.473	52	24010	73
being	49120	82	0.483	58	23875	75
however	48920	83	0.489	62	21728	89
them	45552	84	0.526	79	20328	92
before	45321	85	0.516	77	23203	78
second	44938	86	0.573	98	22246	85
well	44153	87	0.499	67	23319	77
since	43408	88	0.535	82	22796	83
until	42038	89	0.555	91	22846	82
called	40221	90	0.548	86	20387	91
several	40194	91	0.529	81	21768	88
following	39075	92	0.570	94	21800	87
early	38268	93	0.552	89	19625	93
same	34850	94	0.548	85	19047	94
so	34829	95	0.555	90	16941	96
any	32853	96	0.571	96	16559	97
because	32510	97	0.568	93	15997	98
now	32085	98	0.589	100	18433	95
although	29261	99	0.571	97	15764	100
another	28199	100	0.564	92	15986	99

E.2 PORTUGUESE

Table E.2: Top 100 Portuguese 1-grams

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
de	of	224954	1	0.139	1	9710	1
a	the	114312	2	0.158	3	7711	4

Table E.2 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
e	and	106238	3	0.142	2	7476	6
o	the	93771	4	0.180	4	6485	11
em	in	76849	5	0.187	5	7985	3
do	of the	67897	6	0.219	6	7097	8
da	of the	61418	7	0.220	7	7398	7
que	that	52467	8	0.236	9	5072	17
no	in the	37004	9	0.240	11	6481	12
um	a, one	36434	10	0.240	10	6818	10
com	with	34216	11	0.225	8	5209	16
uma	a, one	33270	12	0.260	14	6888	9
para	to	31053	13	0.267	15	4209	18
é	is	30656	14	0.352	25	8338	2
na	in the	29529	15	0.249	12	5902	13
por	by	26011	16	0.251	13	5467	14
os	the (pl.)	25443	17	0.302	18	3919	22
foi	was	25033	18	0.330	22	5447	15
como	how	22237	19	0.289	16	4133	19
dos	of the	17951	20	0.309	19	4059	21
as	the	17129	21	0.329	21	3494	25
se	if	14659	22	0.334	23	3068	27
ao	to the	14498	23	0.297	17	3658	23
mais	more	13681	24	0.324	20	3207	26
sua	yours	13668	25	0.372	26	3536	24
não	no, not	11731	26	0.379	28	2641	33
seu	his, yours	11445	27	0.407	32	2964	30
são	are	11165	28	0.488	44	2633	34
das	of the	10702	29	0.378	27	3027	28
também	too	10608	30	0.339	24	4061	20
ou	or	9118	31	0.520	59	2468	36
à	to the	8821	32	0.403	31	2969	29
pela	by the	8413	33	0.381	29	2920	31

Table E.2 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
referências	references	7809	34	0.602	94	7640	5
pelo	by the	7753	35	0.393	30	2677	32
ser	be	7304	36	0.421	34	2318	37
entre	between	7269	37	0.407	33	2478	35
anos	years	6865	38	0.478	40	2203	39
mas	but	5861	39	0.435	35	1903	43
era	was	5700	40	0.498	50	2079	40
nos	in the	5380	41	0.449	36	2209	38
até	until	4916	42	0.451	37	1948	42
foram	were	4780	43	0.491	45	1690	49
seus	yours	4754	44	0.471	39	1850	45
sobre	about	4661	45	0.502	52	1894	44
quando	when	4297	46	0.469	38	1729	48
tem	has	4284	47	0.556	73	2072	41
grande	big	4260	48	0.508	53	1636	53
onde	where	4235	49	0.497	49	1844	46
ano	year	4190	50	0.565	78	1599	58
nas	in the	3992	51	0.479	41	1778	47
durante	during	3944	52	0.520	58	1665	51
depois	after	3864	53	0.534	64	1574	59
após	after	3817	54	0.536	66	1538	60
primeiro	first	3812	55	0.529	62	1654	52
sendo	being	3793	56	0.496	47	1631	54
mesmo	same	3750	57	0.485	43	1521	61
primeira	first	3698	58	0.527	61	1610	57
parte	part	3527	59	0.510	54	1627	56
dois	two	3490	60	0.511	56	1490	62
ainda	still	3397	61	0.495	46	1427	67
suas	yours	3363	62	0.497	48	1464	64
maior	greatest	3345	63	0.568	80	1270	74
outros	others	3337	64	0.483	42	1483	63

Table E.2 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
história	history	3260	65	0.547	69	1630	55
segundo	second	3253	66	0.577	84	1669	50
aos	to the	3239	67	0.510	55	1440	66
já	already	3199	68	0.500	51	1409	69
está	is	2967	69	0.604	98	1419	68
nome	name	2800	70	0.607	99	1449	65
apenas	only	2796	71	0.522	60	1331	70
muito	very	2735	72	0.553	71	1205	79
vez	time	2733	73	0.520	57	1199	80
três	three	2729	74	0.540	67	1311	72
nova	new	2561	75	0.602	95	1245	76
ter	have	2510	76	0.533	63	1216	78
este	this	2501	77	0.572	82	1297	73
além	beyond	2498	78	0.541	68	1179	82
desde	since	2453	79	0.564	77	1256	75
então	then	2365	80	0.556	74	1139	85
todos	all	2364	81	0.534	65	1163	83
outras	others	2345	82	0.572	83	1325	71
forma	form	2324	83	0.586	88	1074	91
tempo	time	2321	84	0.567	79	1069	93
sido	been	2302	85	0.579	87	1180	81
duas	two	2296	86	0.555	72	1219	77
assim	this way	2276	87	0.549	70	1134	86
esta	this	2144	88	0.594	92	1156	84
sem	without	2134	89	0.561	76	1070	92
qual	what	2094	90	0.591	91	1129	87
alguns	some	2050	91	0.569	81	1057	94
teve	had	1997	92	0.604	97	1124	88
às	to the	1972	93	0.577	85	984	95
pelos	by the	1934	94	0.557	75	1089	89
antes	before	1878	95	0.578	86	1082	90

Table E.2 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
enquanto	while	1872	96	0.589	89	920	99
partir	leave	1843	97	0.601	93	976	96
bem	good	1700	98	0.589	90	963	97
início	start	1616	99	0.611	100	923	98
outro	other	1403	100	0.604	96	807	100

E.3 JAPANESE

Table E.3: Top 100 Japanese 1-grams

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
なり	become	1934	1	0.591	93	1040	99
出	out	2108	2	0.605	100	1041	98
それ	it	2112	3	0.571	80	1057	97
でき	can	2227	4	0.603	98	1030	100
後に	later	2248	5	0.586	88	1258	93
んで	why	2265	6	0.587	89	1153	96
しかし	however	2441	7	0.599	94	1210	95
多く	many	2539	8	0.603	97	1246	94
受け	received	2815	9	0.572	82	1282	92
その後	after that	2847	10	0.589	90	1495	82
め	because	2873	11	0.538	71	1399	86
け	over	2881	12	0.564	79	1421	85
ま	up	2993	13	0.601	96	1358	89
前	before	3006	14	0.589	91	1349	90
見	you see	3173	15	0.560	78	1313	91
なか	naka	3350	16	0.547	75	1440	84
じ	tooth	3360	17	0.579	85	1536	80
上	up	3364	18	0.544	73	1504	81
なく	without	3433	19	0.494	58	1470	83
脚注	footnote	3528	20	0.572	81	3520	28

Table E.3 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
時	time	3573	21	0.586	87	1363	88
ず	to	3634	22	0.493	57	1565	78
化	conversion into	3759	23	0.605	99	1399	87
これ	this	3824	24	0.533	69	1542	79
り	ri	3825	25	0.544	74	1569	77
関連	relation	3827	26	0.574	83	3165	36
でも	but	3847	27	0.493	56	1703	74
られる	to be	3862	28	0.542	72	1635	76
外部	outside	3976	29	0.574	84	3873	25
現在	current	3999	30	0.601	95	1904	65
き	can	4028	31	0.495	59	1800	71
ん	hmm	4029	32	0.557	77	1868	67
リンク	link	4103	33	0.580	86	3869	26
名	name	4138	34	0.590	92	1800	72
え	huh	4376	35	0.482	50	1873	66
によって	by	4377	36	0.552	76	1658	75
われ	i	4712	37	0.492	55	1858	68
ば	the	4729	38	0.486	51	1830	69
おり	cage	4806	39	0.491	53	1830	70
により	by	4834	40	0.519	67	1995	64
せ	to	4877	41	0.490	52	1773	73
後	rear	5013	42	0.502	61	2163	57
より	than	5214	43	0.504	62	2083	59
く	to	5233	44	0.471	48	2076	61
大	big	5281	45	0.525	68	2048	63
による	by	5316	46	0.497	60	2080	60
行	line	6359	47	0.508	65	2206	56
もの	thing	6451	48	0.491	54	2055	62
また	also	6710	49	0.405	32	2395	51
あり	there	6851	50	0.432	38	2472	47
中	during	6943	51	0.422	36	2466	49

Table E.3 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
まで	until	7328	52	0.432	39	2610	44
その	that	7387	53	0.439	42	2344	54
よう	yo	7440	54	0.452	45	2121	58
あっ	ah	7571	55	0.451	44	2530	46
だ	it is	7886	56	0.441	43	2472	48
という	to say	8131	57	0.453	46	2260	55
なる	become	8268	58	0.422	35	2643	41
へ	what	8301	59	0.437	41	2640	42
れる	to be	8687	60	0.427	37	2594	45
人	man	8817	61	0.504	63	2362	53
か	or	8979	62	0.402	31	2695	39
者	a person	9360	63	0.507	64	2416	50
的	target	10160	64	0.466	47	2386	52
この	this	10250	65	0.436	40	2679	40
ため	for	10303	66	0.385	29	2812	38
ない	absent	11327	67	0.407	33	2627	43
日本	japan	11444	68	0.537	70	3205	34
など	such	12132	69	0.416	34	3049	37
てい	listen	13080	70	0.361	26	3183	35
ら	et al	13757	71	0.345	23	3421	29
い	there	14536	72	0.332	22	3639	27
や	ya	15487	73	0.377	28	3331	32
って	what?	15864	74	0.346	24	3389	30
では	then.	16344	75	0.368	27	3337	31
として	as	17310	76	0.303	18	4158	23
こと	about	18144	77	0.359	25	3304	33
っ	combining form	24871	78	0.313	20	4072	24
から	from	25825	79	0.259	13	4733	18
も	also	27453	80	0.278	15	4266	22
する	to	30891	81	0.292	17	4701	19
ある	is there	31594	82	0.315	21	5255	14

Table E.3 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
いる	to have	32254	83	0.309	19	4598	21
な	what	34213	84	0.259	14	4765	17
した	did	35635	85	0.281	16	4895	16
日	day	39943	86	0.512	66	4642	20
し	to	43498	87	0.215	8	5415	11
さ	to	46015	88	0.256	12	5322	12
れ	re	49463	89	0.247	11	5301	13
月	month	50619	90	0.475	49	5090	15
て	the	51565	91	0.233	9	5447	10
と	when	77661	92	0.185	5	6045	7
で	so	85980	93	0.181	4	6303	5
た	it was	87708	94	0.239	10	5866	9
年	year	100903	95	0.398	30	6212	6
が	but	112993	96	0.199	7	6045	8
は	is	121900	97	0.178	3	7021	1
を	to	136517	98	0.190	6	6331	4
に	into	152924	99	0.148	2	6692	3
の	of	277146	100	0.113	1	7003	2

E.4 HEBREW

Table E.4: Top 100 Hebrew 1-grams

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
של	of	32369	1	0.153	1	2254	1
את	you, the	18220	2	0.218	2	1971	5
על	on, about	15971	3	0.205	6	2006	4
הוא	he	8023	4	0.353	3	1766	6
ב	in	7154	5	0.446	11	1444	8
עם	also	6538	6	0.317	4	1548	7
בשנת	in the year	4511	7	0.498	15	1260	11

Table E.4 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
היה	was	4157	8	0.430	16	1183	15
באתר	place	4157	9	0.571	33	1280	10
גם	also	4113	10	0.326	5	1382	9
לא	no	4107	11	0.408	23	1069	19
היא	she	3814	12	0.482	12	1182	16
לאחר	after	3790	13	0.409	7	1186	14
בין	between	3573	14	0.377	14	1215	12
ידי	by me	3310	15	0.409	10	1195	13
ה	God	3303	16	0.506	26	1028	20
או	or	2917	17	0.522	28	866	25
כי	because	2639	18	0.538	43	697	37
זה	it	2614	19	0.417	13	1015	21
אך	but	2297	20	0.476	19	910	23
עד	until	2274	21	0.433	20	990	22
יותר	more	2115	22	0.462	84	804	27
כל	all	2090	23	0.448	17	884	24
הייתה	was	2075	24	0.515	18	792	28
חיצוניים	external	2037	25	0.420	8	2008	2
אשר	which	2033	26	0.588	87	705	36
קישורים	links	2017	27	0.425	9	2008	3
בית	home	2016	28	0.662	81	601	44
זו	this	1921	29	0.502	51	768	30
הם	they	1788	30	0.529	22	733	33
הברית	United States	1780	31	0.661	82	576	52
אל	to	1702	32	0.602	42	574	53
ביותר	most	1695	33	0.540	34	733	34
הראשון	the first one	1615	34	0.540	29	779	29
בו	at him	1567	35	0.494	38	739	32
היו	there were	1514	36	0.556	31	587	47
שם	name	1493	37	0.551	41	829	26
אחד	one	1487	38	0.494	57	741	31

Table E.4 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
כמו	same as	1462	39	0.523	39	726	35
כאשר	when	1421	40	0.562	54	606	43
בן	son	1395	41	0.646	65	586	48
שלו	his	1333	42	0.610	95	618	42
במהלך	during	1324	43	0.574	48	620	41
מספר	a number	1318	44	0.549	47	653	39
כך	so	1314	45	0.511	21	632	40
כדי	in order to	1309	46	0.583	37	548	60
אותו	him	1251	47	0.554	30	598	45
העולם	the world	1241	48	0.631	83	580	51
שני	crimson	1213	49	0.559	25	667	38
לו	to him	1160	50	0.599	50	539	64
הערות	remarks	1157	51	0.517	40	1135	17
אף	nose	1149	52	0.540	24	556	57
שוליים	margins	1137	53	0.518	36	1132	18
בה	in her	1134	54	0.575	35	569	55
הראשונה	the first	1121	55	0.620	49	572	54
זאת	this	1115	56	0.528	27	558	56
יש	there is	1108	57	0.637	71	519	68
רבים	many	1097	58	0.561	32	546	62
שנה	year	1094	59	0.632	78	593	46
ל	to	1075	60	0.644	80	520	67
שלא	not	1071	61	0.648	76	399	87
פי	times	1062	62	0.589	68	487	71
רק	only	1021	63	0.545	59	540	63
שנים	years	1014	64	0.584	44	551	58
אם	if	1013	65	0.618	55	423	81
החל	apply	999	66	0.603	46	581	50
השנייה	the second	984	67	0.649	58	533	66
מכן	then	958	68	0.613	64	582	49
ו	and	938	69	0.662	86	497	70

Table E.4 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
חלק	part	934	70	0.609	75	539	65
בשם	on behalf of	929	71	0.628	66	547	61
כן	yes	905	72	0.589	45	549	59
אחת	one	849	73	0.613	61	481	72
השני	the second	849	74	0.641	99	464	75
לפני	before	842	75	0.611	60	467	74
אותה	her	831	76	0.624	63	450	77
מ	m	824	77	0.664	94	472	73
באופן	in a manner	803	78	0.609	70	449	78
בכל	in all	759	79	0.616	53	415	84
להיות	to be	747	80	0.618	73	426	80
דבר	nothing	734	81	0.618	74	390	91
בעקבות	following	688	82	0.673	97	448	79
אחר	other	659	83	0.648	52	401	85
נוספת	another	652	84	0.613	56	508	69
והוא	and he	645	85	0.659	67	455	76
בהם	in them	640	86	0.631	69	423	82
באותה	the same	635	87	0.675	92	421	83
שונים	different	631	88	0.662	91	386	92
ללא	with no	630	89	0.640	88	377	93
ולא	and no	628	90	0.652	79	342	99
עוד	more	621	91	0.635	72	399	86
בשל	mature	621	92	0.670	93	374	94
מה	what	620	93	0.666	85	350	98
רבות	many	609	94	0.652	62	398	89
בנוסף	additionally	600	95	0.668	90	396	90
במשך	during	593	96	0.671	100	399	88
אחרים	others	573	97	0.657	77	366	95
שהיה	was	570	98	0.658	98	350	97
זמן	time	545	99	0.665	89	356	96
ואף	and even	457	100	0.669	96	330	100

E.5 PERSIAN

Table E.5: Top 100 Persian 1-grams

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
در	at	51962	1	0.096	1	2525	1
و	and	51683	2	0.115	4	1840	4
به	to	34849	3	0.113	3	2165	3
از	from	32283	4	0.106	2	1649	6
که	that	20381	5	0.138	5	1753	5
را	take	16509	6	0.223	8	781	14
این	this	15565	7	0.168	6	1253	8
است	is	12067	8	0.275	13	2308	2
با	with	12000	9	0.169	7	932	12
سال	year	7326	10	0.283	15	1062	9
آن	it's	6716	11	0.223	9	1028	10
بود	was	6427	12	0.342	23	702	16
شد	became	6327	13	0.304	18	760	15
یک	one	5835	14	0.330	21	1506	7
برای	to	5580	15	0.276	14	595	21
کرد	made	4683	16	0.344	24	590	22
خود	yourself	4370	17	0.305	19	484	26
بر	on	4342	18	0.237	11	564	24
تا	until the	4088	19	0.230	10	630	18
نیز	also	3467	20	0.273	12	470	28
دارد	has it	3109	21	0.402	51	990	11
شده	have been	2969	22	0.303	17	669	17
می‌شود	gets	2950	23	0.420	60	503	25
نام	name	2945	24	0.369	34	605	20
پس	so	2864	25	0.348	28	434	31
یا	or	2667	26	0.434	74	474	27
شده‌است	has been	2510	27	0.441	83	895	13
دو	two	2492	28	0.317	20	441	29

Table E.5 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
قرار	put	2354	29	0.348	27	617	19
عنوان	title	2149	30	0.359	31	437	30
یکی	one	2055	31	0.338	22	567	23
دیگر	the other	1928	32	0.286	16	374	35
می‌کند	he does	1784	33	0.434	73	362	37
اما	but	1758	34	0.381	42	335	39
هم	both	1736	35	0.359	30	370	36
آن‌ها	they are	1655	36	0.368	33	276	63
وجود	existence	1582	37	0.409	52	320	45
همچنین	also	1555	38	0.346	25	381	33
توسط	by	1554	39	0.376	40	381	34
تاریخ	date	1542	40	0.436	78	423	32
بعد	later	1509	41	0.390	45	310	47
زمان	time	1486	42	0.347	26	321	44
هر	any	1469	43	0.361	32	333	40
داشت	had	1462	44	0.378	41	302	51
داد	gave	1342	45	0.410	53	298	53
مورد	case	1333	46	0.397	48	337	38
دست	hand	1293	47	0.371	36	309	49
کار	work	1289	48	0.357	29	324	43
مردم	people	1275	49	0.441	82	248	76
بودند	they were	1195	50	0.427	70	236	84
کند	slowly	1162	51	0.459	99	262	67
بین	among	1144	52	0.389	44	300	52
بزرگ	the big	1131	53	0.424	67	282	57
روی	on	1128	54	0.416	58	312	46
پیش	before	1117	55	0.375	38	280	60
صورت	the face	1082	56	0.421	61	332	41
میان	between	1071	57	0.415	56	258	69
روز	day	1048	58	0.444	87	249	74
کرده	done	1029	59	0.425	69	310	48

Table E.5 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
اول	first	1006	60	0.445	89	278	62
کردند	they made	992	61	0.400	49	236	83
نظر	opinion	990	62	0.382	43	278	61
بیشتر	more	986	63	0.402	50	286	55
آغاز	the beginning	986	64	0.415	55	267	65
گرفت	took	983	65	0.395	47	260	68
سه	three	943	66	0.373	37	304	50
دوم	second	926	67	0.423	64	252	72
برخی	some	914	68	0.411	54	240	81
مانند	as	913	69	0.451	93	245	78
انجام	do	913	70	0.453	94	288	54
شدند	they were	912	71	0.422	62	209	93
شود	to be	896	72	0.451	92	285	56
ولی	but	875	73	0.456	98	228	89
بسیار	very	847	74	0.415	57	264	66
بار	bar	838	75	0.438	79	281	59
تنها	single	811	76	0.424	65	247	77
دلیل	the reason	788	77	0.423	63	248	75
بسیاری	many	787	78	0.371	35	233	86
حال	now	764	79	0.431	71	273	64
چند	how many	754	80	0.375	39	254	71
زیر	under	739	81	0.456	97	330	42
جمله	sentence	727	82	0.459	100	240	82
راه	the way	709	83	0.436	77	236	85
همراه	along	693	84	0.443	85	282	58
داده	data	691	85	0.451	91	252	73
شدن	become	676	86	0.456	96	245	79
بیش	more	670	87	0.440	81	231	88
پایان	end	662	88	0.444	88	225	90
همین	this	651	89	0.393	46	241	80
یافت	found	646	90	0.436	76	231	87

Table E.5 continued from previous page

Token	Translation	Abs. Freq	# Freq	DP	# DP	Disp	# Disp
گرفته	taken	626	91	0.433	72	258	70
می‌شد	been	591	92	0.424	66	192	98
رسید	receipt	569	93	0.451	90	201	96
تحت	under	558	94	0.424	68	201	95
سر	head	555	95	0.418	59	220	91
توجه	attention	551	96	0.435	75	218	92
همان	same	538	97	0.454	95	206	94
چهار	four	537	98	0.441	84	200	97
جای	instead	466	99	0.443	86	183	99
کنار	next to the	452	100	0.439	80	172	100

BIBLIOGRAPHY

- Adelman, James S, Gordon DA Brown, and José F Quesada (2006). "Contextual diversity, not word frequency, determines word-naming and lexical decision times." In: *Psychological Science* 17.9, pp. 814–823 (cit. on pp. 23, 24).
- Anderson, Jonathan (1983). "Lix and rix: Variations on a little-known readability index." In: *Journal of Reading* 26.6, pp. 490–496 (cit. on p. 37).
- ATOS for Text, ATOS for Books* (n.d.). <https://www.renaissance.com/products/practice/accelerated-reader-360/atos-and-text-complexity/>. Accessed: 2018-08-14 (cit. on p. ix).
- Aziz, Anealka, Chan Yuen Fook, and Zubaida Alsree (2010). "Computational Text Analysis: A More Comprehensive Approach to Determine Readability of Reading Materials." In: *Advances in Language and Literary Studies* 1.2, pp. 200–219 (cit. on p. 18).
- Bailin, Alan and Ann Grafstein (2001). "The linguistic assumptions underlying readability formulae: A critique." In: *Language & Communication* 21.3, pp. 285–301 (cit. on pp. 7, 11, 18, 33).
- Baker, Lynda M, Feleta L Wilson, and Marge Kars (1997). "The readability of medical information on InfoTrac: Does it meet the needs of people with low literacy skills?" In: *Reference & User Services Quarterly*, pp. 155–160 (cit. on pp. ix, x, 11).
- Baker, Paul (2003). *Polari - the lost language of gay men*. Routledge (cit. on p. 7).
- Bauer, Laurie and Paul Nation (1993). "Word families." In: *International journal of Lexicography* 6.4, pp. 253–279 (cit. on pp. 47, 48).
- Begeny, John C and Diana J Greene (2014). "Can readability formulas be used to successfully gauge difficulty of reading materials?" In: *Psychology in the Schools* 51.2, pp. 198–215 (cit. on pp. ix, x, 11–13, 18).

- Benjamin, Rebekah George (2012). "Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty." In: *Educational Psychology Review* 24.1, pp. 63–88 (cit. on pp. 1, 11, 14, 38).
- Biderman, Maria Tereza Camargo (1999). "Conceito linguístico de palavra." In: *paLavra* 5 (cit. on p. 53).
- Bjork, Robert A (1994). "Memory and metamemory considerations in the training of human beings." In: *Metacognition: Knowing about knowing* 185 (cit. on p. 6).
- Björnsson, Carl-Hugo (1983). "Readability of newspapers in 11 languages." In: *Reading Research Quarterly*, pp. 480–497 (cit. on p. 37).
- Boerma, Tessel et al. (July 2017). "Grammatical Morphology in Monolingual and Bilingual Children With and Without Language Impairment: The Case of Dutch Plurals and Past Participles." In: *Journal of Speech Language and Hearing Research* 60.7, p. 2064. DOI: [10.1044/2017_jslhr-1-16-0351](https://doi.org/10.1044/2017_jslhr-1-16-0351) (cit. on p. 9).
- Bormuth, John R (1971). "Development of Standards of Readability: Toward a Rational Criterion of Passage Performance. Final Report." In: URL: <https://files.eric.ed.gov/fulltext/ED054233.pdf> (cit. on pp. 13, 14).
- Bringhurst, Robert (2004). *The elements of typographic style*. Hartley & Marks Vancouver, British Columbia (cit. on p. xi).
- Brown, Ronan, Rob Waring, and Sangrawee Donkaewbua (2008). "Incidental vocabulary acquisition from reading, reading-while-listening, and listening to stories." In: *Reading in a foreign language* 20.2, p. 136 (cit. on p. 49).
- Brysaert, Marc and Boris New (2009). "Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English." In: *Behavior research methods* 41.4, pp. 977–990 (cit. on pp. 23, 25, 41, 51, 52, 58, 77, 120).
- Bu, Fan, Xiao-Yan Zhu, and Ming Li (2011). "A new multiword expression metric and its applications." In: *Journal of Computer Science and Technology* 26.1, pp. 3–13 (cit. on pp. ix, x, 27).

- Carroll, Sean M and Jennifer Chen (2004). "Spontaneous Inflation and the Origin of the Arrow of Time." In: *arXiv preprint hep-th/0410270* (cit. on p. 7).
- Caseli, Helena Medeiros de et al. (Apr. 2010). "Alignment-based extraction of multiword expressions." In: *Language Resources and Evaluation* 44.1, pp. 59–77. ISSN: 1574-0218. DOI: [10.1007/s10579-009-9097-9](https://doi.org/10.1007/s10579-009-9097-9). URL: <https://doi.org/10.1007/s10579-009-9097-9> (cit. on p. 27).
- Chall, Jeanne Sternlicht and Edgar Dale (1948). "A formula for predicting readability: Instructions." In: *Educational research bulletin*, pp. 37–54 (cit. on p. 12).
- (1995). *Readability revisited: The new Dale-Chall readability formula*. Brookline Books (cit. on p. 12).
- Coleman, Meri and Ta Lin Liao (1975). "A computer readability formula designed for machine scoring." In: *Journal of Applied Psychology* 60.2, p. 283 (cit. on p. 37).
- Cook, Vivian (2008). *Second Language Learning and Language Teaching*. ISBN: 978 0 340 95876 6. DOI: [10.4324/9780203770511](https://doi.org/10.4324/9780203770511) (cit. on p. 1).
- Davanzo, Christopher (2016). "The Supplemental Use of Graded Reader Sets in the Extended Reading Classroom." In: *Journal of the Nanzan Academic Society* 99, pp. 213–219 (cit. on p. 1).
- Deslauriers, Louis et al. (2012). "Transforming the lowest-performing students: an intervention that worked." In: *Journal of College Science Teaching* 41.6, p. 76 (cit. on p. 6).
- Diller, Karl Conrad (1978). *The language teaching controversy*. Newbury House Publishers (cit. on p. 8).
- Douglas, Scott Roy (2010). "Non-native English speaking students at university: Lexical richness and academic success." PhD thesis. University of Calgary (cit. on p. 99).
- Du Bois, John W et al. (2000). *Santa Barbara Corpus of Spoken American English* (cit. on p. 98).
- Duffy, Thomas M. and Paula Kabance (1982). "Testing a readable writing approach to text revision." In: 74, pp. 733–748. ISSN: 0022-0663. DOI: [10.1037/0022-0663.74.5.733](https://doi.org/10.1037/0022-0663.74.5.733) (cit. on p. 12).
- Evans, Nicholas and Stephen C Levinson (2009). "The myth of language universals: Language diversity and its importance for cognitive science." In: *Behavioral and brain sciences* 32.5, pp. 429–448 (cit. on p. 9).

- Flesch, Rudolph (1948). "A new readability yardstick." In: *Journal of applied psychology* 32.3, p. 221 (cit. on pp. ix, x, 1, 11).
- Friedman, Daniela B. et al. (June 2009). "African American Men's Understanding and Perceptions About Prostate Cancer: Why Multiple Dimensions of Health Literacy are Important in Cancer Communication." In: *Journal of Community Health* 34.5, pp. 449–460. DOI: [10.1007/s10900-009-9167-3](https://doi.org/10.1007/s10900-009-9167-3) (cit. on p. 14).
- Geeraerts, Dirk (2010). *Theories of lexical semantics*. Oxford University Press (cit. on p. 31).
- Goulden, Robin, Paul Nation, and John Read (1990). "How large can a receptive vocabulary be?" In: *Applied linguistics* 11.4, pp. 341–363 (cit. on p. 8).
- Granger, Sylviane (2014). "A lexical bundle approach to comparing languages: Stems in English and French." In: *Languages in Contrast* 14.1, pp. 58–72 (cit. on p. 28).
- Gries, Stefan Th. (n.d.). "Analyzing dispersion." In: *Practical handbook of corpus linguistics*. Ed. by Magali Paquot & Stefan Th. Gries. Berlin & New York: Springer. Chap. Practical handbook of corpus linguistics. URL: http://www.linguistics.ucsb.edu/faculty/stgries/research/ToApp_STG_Dispersion_PHCL.pdf. Forthcoming (cit. on pp. 23, 76).
- (2008). "Dispersions and adjusted frequencies in corpora." In: *International journal of corpus linguistics* 13.4, pp. 403–437 (cit. on pp. ix, x, 75–78).
- (2010). "Dispersions and adjusted frequencies in corpora: further explorations." In: *Corpus linguistic applications: current studies, new directions*, pp. 197–212 (cit. on pp. ix, x, 75, 76).
- Gries, Stefan Th. and Nick C. Ellis (2015). "Statistical Measures for Usage-Based Linguistics." In: *Language Learning* 65, pp. 228–255. ISSN: 0023-8333. DOI: [10.1111/lang.12119](https://doi.org/10.1111/lang.12119) (cit. on pp. 24, 75).
- Griffiths, T. L. (2011). "Rethinking language: How probabilities shape the words we use." In: *Language Learning* 108, pp. 3825–3826. ISSN: 0027-8424. DOI: [10.1073/pnas.1100760108](https://doi.org/10.1073/pnas.1100760108) (cit. on p. 17).
- Grundner, TM (1980). "On the readability of surgical consent forms." In: *New England Journal of Medicine* 302.16, pp. 900–902 (cit. on p. 11).

- Hancke, Julia, Sowmya Vajjala, and Detmar Meurers (2012). “Readability classification for German using lexical, syntactic, and morphological features.” In: *Proceedings of COLING 2012*, pp. 1063–1080 (cit. on pp. 93, 94).
- Hayes, Donald P. (1988). “Speaking and writing: Distinct patterns of word choice.” In: 27, pp. 572–585. ISSN: 0749-596X. DOI: [10.1016/0749-596x\(88\)90027-7](https://doi.org/10.1016/0749-596x(88)90027-7) (cit. on p. 97).
- Hills, Thomas T. et al. (2010). “The associative structure of language: Contextual diversity in early word learning.” In: 63, pp. 259–273. ISSN: 0749-596X. DOI: [10.1016/j.jml.2010.06.002](https://doi.org/10.1016/j.jml.2010.06.002) (cit. on p. 25).
- Hughes, Michael G et al. (2013). “Learner-controlled practice difficulty in the training of a complex task: Cognitive and motivational mechanisms.” In: *Journal of Applied Psychology* 98.1, p. 80 (cit. on p. 6).
- Jaszczolt, K. M. (Sept. 11, 2005). *Default Semantics: Foundations of a Compositional Theory of Acts of Communication*. Oxford University Press. 304 pp. ISBN: 0199261989. URL: https://www.ebook.de/de/product/3616863/k_m_jaszczolt_default_semantics_foundations_of_a_compositional_theory_of_acts_of_communication.html (cit. on p. 22).
- Kandula, Sasikiran, Dorothy Curtis, and Qing Zeng-Treitler (2010). “A semantic and syntactic text simplification tool for health content.” In: *AMIA annual symposium proceedings*. Vol. 2010. American Medical Informatics Association, p. 366 (cit. on pp. 18, 21).
- Kwary, Deny A. (2018). “A corpus and a concordancer of academic journal articles.” In: 16, pp. 94–100. ISSN: 2352-3409. DOI: [10.1016/j.dib.2017.11.023](https://doi.org/10.1016/j.dib.2017.11.023) (cit. on pp. 98, 104).
- Leroy, Gondy and David Kauchak (2014). “The effect of word familiarity on actual and perceived text difficulty.” In: *Journal of the American Medical Informatics Association* 21.e1, e169–e172 (cit. on pp. ix, x, 6, 12, 15, 18–20, 22, 75).
- Leroy, Gondy, David Kauchak, and Obay Mouradi (2013). “A user-study measuring the effects of lexical simplification and coherence enhancement on perceived and actual text difficulty.” In: *International Journal of Medical Informatics* 82.8,

- pp. 717–730. ISSN: 1386-5056. DOI: [10.1016/j.ijmedinf.2013.03.001](https://doi.org/10.1016/j.ijmedinf.2013.03.001). URL: <http://www.sciencedirect.com/science/article/pii/S1386505613000592> (cit. on pp. 19, 21).
- Lopopolo, Alessandro et al. (2017). “Using stochastic language models (SLM) to map lexical, syntactic, and phonological information processing in the brain.” In: *PloS One* 12.5, e0177794 (cit. on p. 21).
- Marc Brysbaert Michaël Stevens, Paweł Mandera and Emmanuel Keuleers (2016). “How Many Words Do We Know? Practical Estimates of Vocabulary Size Dependent on Word Definition, the Degree of Language Input and the Participant’s Age.” In: *Frontiers in Psychology* 7 (cit. on pp. 47, 50, 51).
- Martinez-Gómez, Pascual and Akiko Aizawa (2013). “Diagnosing causes of reading difficulty using Bayesian networks.” In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 1383–1391 (cit. on pp. 21, 22, 93, 94).
- Martinez, Ron and Victoria A Murphy (2011). “Effect of frequency and idiomaticity on second language reading comprehension.” In: *TESOL Quarterly* 45.2, pp. 267–290 (cit. on pp. 7, 28, 29).
- Martinez, Ron and Norbert Schmitt (2012). “A phrasal expressions list.” In: *Applied linguistics* 33.3, pp. 299–320 (cit. on pp. 27, 28).
- Massachusetts Institute of Technology, MIT (n.d.). *Specifications for Thesis Preparation*. URL: <https://libraries.mit.edu/archives/thesis-specs/thesis-specs-2016.pdf> (cit. on p. xi).
- Maylath, Bruce (1997). “Why Do They Get It When I Say “Gingivitis” But Not When I Say “Gum Swelling?”” In: *New Directions for Teaching and Learning* 1997.70, pp. 29–37. ISSN: 1536-0768. DOI: [10.1002/tl.7003](https://doi.org/10.1002/tl.7003) (cit. on p. 7).
- Meade, Cathy D and Cyrus F Smith (1991). “Readability formulas: cautions and criteria.” In: *Patient education and counseling* 17.2, pp. 153–158 (cit. on pp. xi, 11).

- Miltsakaki, Eleni and Audrey Troutt (2007). "Read-x: Automatic evaluation of reading difficulty of web text." In: *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. Association for the Advancement of Computing in Education (AACE), pp. 7280–7286 (cit. on pp. ix, x, 37).
- Murphy, M Lynne (2003). *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms*. Cambridge University Press (cit. on p. 31).
- Nation, I (2006). "How large a vocabulary is needed for reading and listening?" In: *Canadian modern language review* 63.1, pp. 59–82 (cit. on pp. 8, 97).
- Nation, Paul (1997). "The language learning benefits of extensive reading." In: *Language Teacher, Kyoto* 21, pp. 13–16 (cit. on pp. 8, 10).
- Nation, Paul and James Coady (1988). "Vocabulary and reading." In: *Vocabulary and language teaching* 97, p. 110 (cit. on pp. 8, 10, 31).
- Nunberg, Geoffrey (1978). *The pragmatics of reference*. Indiana University Linguistics Club (cit. on p. 31).
- O'Loughlin, Richard (2012). "Tuning in to vocabulary frequency in coursebooks." In: *RELC Journal* 43.2, pp. 255–269 (cit. on p. 9).
- Okamoto, Mayumi (2015). "Is corpus word frequency a good yardstick for selecting words to teach? Threshold levels for vocabulary selection." In: *System* 51, pp. 1–10 (cit. on pp. 1, 23, 25, 75).
- Omidian, Taha, Hesamoddin Shahriari, and Behzad Ghonsooly (2017). "Evaluating the Pedagogic Value of Multi-Word Expressions Based on EFL Teachers' and Advanced Learners' Value Judgments." In: *TESOL Journal* 8.2, pp. 489–511 (cit. on p. 27).
- Perea, Manuel, Ana Paula Soares, and Montserrat Comesaña (2013). "Contextual diversity is a main determinant of word identification times in young readers." In: *Journal of Experimental Child Psychology* 116.1, pp. 37–44 (cit. on p. 24).
- Pichert, James W. and Peggy Elam (1985). "Readability formulas may mislead you." In: 7, pp. 181–191. ISSN: 0738-3991. DOI: 10.1016/0738-3991(85)90008-4 (cit. on p. 12).

- Pigada, Maria and Norbert Schmitt (2006). "Vocabulary acquisition from extensive reading: A case study." In: *Reading in a foreign language* 18.1, p. 1 (cit. on pp. 10, 31).
- Press, Oxford University (n.d.). *Oxford living dictionaries*. Oxford University Press. URL: <https://en.oxforddictionaries.com/> (cit. on p. 5).
- Pyc, Mary A. and Katherine A. Rawson (May 2009). "Testing the retrieval effort hypothesis: Does greater difficulty correctly recalling information lead to higher levels of memory?" In: *Journal of Memory and Language* 60.4, pp. 437-447. DOI: 10.1016/j.jml.2009.01.004 (cit. on p. 6).
- Raso, Tommaso and Heliana Mello (2012). *C-ORAL-BRASIL I: corpus de referência do português brasileiro falado informal. I*. Portuguese. Editora UFMG, p. 332. ISBN: 978-85-7041-943-9 (cit. on p. 103).
- Razek, Joseph R, Gordon A Hosch, and Daniel Pearl (1982). "Readability of accounting textbooks." In: *The Journal of Business Education* 58.1, pp. 23-26 (cit. on pp. ix, x, 11).
- Readable.io* (n.d.). <https://readable.io/>. Accessed: 2018-08-14 (cit. on pp. ix, x).
- Renandya, Willy A and George M Jacobs (2016). "Extensive Reading and Listening in the L2 Classroom." In: *English Language Teaching Today*. Springer, pp. 97-110 (cit. on p. 10).
- Reynolds, Barry Lee and David Wible (2014). "Frequency in incidental vocabulary acquisition research: An undefined concept and some consequences." In: *TESOL Quarterly* 48.4, pp. 843-861 (cit. on pp. 47, 48).
- Sag, Ivan A et al. (2002). "Multiword expressions: A pain in the neck for NLP." In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pp. 1-15 (cit. on p. 27).
- Saragi, Thomas, Ian Stephen Paul Nation, and Gerold Fritz Meister (1978). "Vocabulary learning and reading." In: *System* 6.2, pp. 72-78 (cit. on p. 10).
- Schmitt, Norbert (2010). *Researching vocabulary: A vocabulary research manual*. Springer (cit. on p. 50).
- Schriver, Karen A. (2000). "Readability formulas in the new millennium: what's the use?" In: 24, pp. 138-140. ISSN: 1527-6805. DOI: 10.1145/344599.344638 (cit. on pp. ix, x, 12).

- Shiffrin, Richard M and Mark Steyvers (1997). “A model for recognition memory: REM—retrieving effectively from memory.” In: *Psychonomic bulletin & review* 4.2, pp. 145–166 (cit. on p. 23).
- Silva, Eduardo (2018). “Análise de textos enciclopédicos da Simple English Wikipedia e da Wikipedia: algumas discussões para o ensino de língua inglesa.” In: *Revista de Estudos da Linguagem* 26.2, pp. 769–792. ISSN: 2237-2083. DOI: [10.17851/2237-2083.26.2.769-792](https://doi.org/10.17851/2237-2083.26.2.769-792). URL: <http://periodicos.letras.ufmg.br/index.php/relin/article/view/12509> (cit. on p. 94).
- Simpson-Vlach, Rita and Nick C Ellis (2010). “An academic formulas list: New methods in phraseology research.” In: *Applied linguistics* 31.4, pp. 487–512 (cit. on p. 28).
- Souza, Ricardo de and Heliana Mello (2007). “Realização argumental na língua do aprendiz de línguas estrangeiras – possibilidades de exploração da interface entre semântica e sintaxe.” In: *Revista Virtual de Estudos da Linguagem – ReVEL* 5.8, pp. 1–19 (cit. on p. 9).
- Spache, George (1953). “A new readability formula for primary-grade reading materials.” In: *The Elementary School Journal* 53.7, pp. 410–413 (cit. on pp. ix, x, 11).
- Stein, Gertrude (1922). “Sacred emily.” In: *Geography and plays*, pp. 178–88 (cit. on p. 47).
- Steyvers, Mark and Kenneth J Malmberg (2003). “The effect of normative context variability on recognition memory.” In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 29.5, p. 760 (cit. on p. 25).
- Sticht, Thomas G (1973). “Research Toward the Design, Development and Evaluation of a Job-Functional Literacy Training Program for the United States Army.” In: *Literacy Discussion* 4.3, pp. 339–69 (cit. on p. 11).
- Sydes, Matthew and James Hartley (1997). “A Thorn in the Flesh: Observations on the Unreliability of Computer-Based Readability Formulae.” In: *British Journal of Educational Technology* 28.2, pp. 143–145 (cit. on pp. ix, x).
- Tim vor der Brück Sven Hartrumpf, Hermann Helbig (2008). “A Readability Checker with Supervised Learning Using Deep Indicators.” In: *Informatica (03505596)* 32.4 (cit. on pp. 38, 93).

- Tweedie, Fiona J and R Harald Baayen (1998). "How variable may a constant be? Measures of lexical richness in perspective." In: *Computers and the Humanities* 32.5, pp. 323–352 (cit. on p. 99).
- UFMG, Colegiado POSLIN (Dec. 2013). *Resolução nº 03/2013*. URL: <http://www.poslin.lettras.ufmg.br/> (cit. on p. xi).
- Van Heuven, Walter JB et al. (2014). "SUBTLEX-UK: A new and improved word frequency database for British English." In: *The Quarterly Journal of Experimental Psychology* 67.6, pp. 1176–1190 (cit. on p. 75).
- Vermeer, Anne (1992). "Exploring the Second Language Learner Lexicon." In: *The construct of language proficiency: Applications of psychological models to language assessment*, p. 147 (cit. on p. 8).
- Ward, Jeremy and Jitlada Chuenjundaeng (2009). "Suffix knowledge: Acquisition and applications." In: *System* 37.3, pp. 461–469 (cit. on p. 49).
- Wikipedia contributors (2018a). *Avoid overly technical language*. In: *Wikipedia: Make technical articles understandable — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-June-2018]. URL: https://en.wikipedia.org/wiki/Wikipedia:Make_technical_articles_understandable#Avoid_overly_technical_language (cit. on p. 42).
- (2018b). *Vocabulary*. In: *Wikipedia: Manual of Style — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-June-2018]. URL: https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style#Vocabulary (cit. on p. 42).
- Zipf, George Kingsley (1949). *Human Behavior and the Principle of Least Effort*. Oxford, England: Addison-Wesley Press (cit. on pp. 17, 32).