UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Estatística
Programa de Pós Graduação em Estatística

Larissa Sayuri Futino Castro dos Santos

Mining Large Amount of Short Text Data in your Desktop

Belo Horizonte
2019

Larissa Sayuri Futino Castro dos Santos

Mining Large Amount of Short Text Data in your Desktop

Versão Final

Tese de Doutorado* apresentada ao Programa de Pós Graduação em Estatística da Universidade Federal de Minas Gerais como requisito parcial para obtenção do título de Doutor em Estatística.
*Pesquisa financiada pela CAPES.

Orientador: Dr. Marcos Oliveira Prates
Orientador internacional: Dr. Michael Oakes

Belo Horizonte
2019

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz - CRB 6ª Reg. nº 819

Santos, Larissa Sayuri Futino Castro dos.

S237m Mining large amount of short text data in your desktop / Larissa Sayuri Futino Castro dos Santos — Belo Horizonte, 2019.
76 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas Gerais – Departamento de Estatística.

Orientador: Marcos Oliveira Prates
Coorientador internacional: Michael Oakes

1. Estatística – Teses. 2. Análise por conglomerados - Teses. 3. Mineração de dados (Computação) – Teses. 4. Classificação de Textos - Teses. 5. Aprendizado do computador – Teses. I. Orientador. II. Coorientador. III. Título.

CDU 519.2(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

ATA DA DEFESA DE TESE DA ALUNA
LARISSA SAYURI FUTINO CASTRO DOS SANTOS

Realizou-se, no dia 27 de junho de 2019, às 13:30 horas, 2029 Icex, da Universidade Federal de Minas Gerais, a 54ª defesa de tese, intitulada *Mining Large Amount of Short Text Data in your Desktop*, apresentada por LARISSA SAYURI FUTINO CASTRO DOS SANTOS, número de registro 2015670062, graduada no curso de ESTATÍSTICA, como requisito parcial para a obtenção do grau de Doutor em ESTATÍSTICA, à seguinte Comissão Examinadora: Prof(a). Marcos Oliveira Prates - Orientador (DEST/UFMG), Prof(a). Ilka Afonso Reis (DEST/UFMG), Prof(a). Marcelo Azevedo Costa (DEST/UFMG), Prof(a). Clécio da Silva Ferreira (DEST/UFJF), Prof(a). Rafael Bassi Stern (DEST/UFSCAR).

A Comissão considerou a tese:

(X) Aprovada

( ) Reprovada

Finalizados os trabalhos, lavrei a presente ata que, lida e aprovada, vai assinada por mim e pelos membros da Comissão.
Belo Horizonte, 27 de junho de 2019.

Prof(a). Marcos Oliveira Prates (Doutor)

Prof(a). Ilka Afonso Reis (Doutora)

Marcelo Azevedo Costa

Prof(a). Marcelo Azevedo Costa (Doutor)

Clécio da S. Ferreira

Prof(a). Clécio da Silva Ferreira (Doutor)

Rafael Bassi Stern

Prof(a). Rafael Bassi Stern (Doutor)

Calma! É aos poucos que a vida vai dando certo.

Desconhecido

Contents

List of Figures

List of Tables

Agradecimentos

Agradeço aos meus pais por terem respeitado a minha escolha e terem apoiado-na e ao meu namorado Douglas pela paciência e conselhos.

Agradeço à Capes pelo fomento desse trabalho e realização do meu sonho de ter uma vivência internacional.

Agradeço ao meu orientador Professor Marcos Prates pela orientação nesses anos.

Agradeço ao meu orientador internacional Professor Michael Oakes e a todos de RGCL pela generosidade e paciência em me mostrarem o novo mundo de NLP.

Agradeço aos professores do Departamento pelos conhecimentos passados.

Agradeço aos servidores e funcionários da UFMG por toda assistência durante esses anos.

Agradeço aos meus colegas de laboratório por toda a generosidade em compartilhar e colaborar.

Agradeço ao Professor Osvaldo Farhat e a todos do São Tomé pela oportunidade de aprender em um ambiente multidisciplinar e colaborativo que ampliou a minha visão de atuação como profissional.

Agradeço aos meus amigos e familiares pelo incentivo constante.

Agradeço à Maio Marketing pelo respeito às minhas ncessidades na realização desse trabalho.

Agradeço ao S4G pela imensa troca de conhecimento que as nossas discussões permitiram.

Resumo

Problemas de classificação/categorização de texto tornam-se ainda mais desafiadores quando os documentos de interesse são curtos. Além da falta de contexto, texto advindos da *web* tem o agravante da espontaneidade, flexibilidade e informalidade. Esse trabalho propõe uma metodologia que viabilize a indução de classificadores de texto para bases de dados grandes por usuários com disponibilidade de computadores comuns e sem conhecimento avançado em computação paralela e/ou distribuída. A metodologia proposta divide-se em dois passos. No primeiro deles, como etapa inicial, procede-se com a partição do banco de dados em subconjuntos de dados menores. No segundo passo cada subconjunto induz um classificador específico a partir de uma técnica supervisionada de Aprendizado de Máquina. A indução de um classificador com a coleção completa é substituída por induções de classificadores com menos dados o que reduz o esforço computacional. Além disso, viabiliza-se também a indução de múltiplos classificadores em distintos cores do computador concomitantemente. Isso denota uma paralelização computacional simples, o que reduz o tempo de processamento para a execução da tarefa. A metodologia também permite o emprego de distintas formas de representação do texto (o uso do vocabulário observado, com diferentes formas de seleção de atributos, o uso de anotação, bigramas, etc). Também é possível o uso de diferentes técnicas de agrupamento e Aprendizado de Máquina. Tais técnicas podem ser especificadas de acordo com as preferências do usuário, contexto e dificuldades do problema ou infra-estrutura disponível. Experimentos com distintos tipo de técnicas de classificação são realizadas. Apresentam-se análises para um base de *tweets* coletados na região de São Paulo-SP, Brasil no tópico de crime. A eficiência da metodologia é comprovada com o seu emprego em uma base de dados de 1.600.000 *tweets* em inglês, no domínio de Análise de Sentimento.

Palavras chave: Classificação de texto, aprendizado de máquina, clustering particional, seleção de atributos, distância Levenshtein/*edit distance.*

Abstract

This work describes the classification of texts as being either crime-related or non crime-related. Given the spontaneity and popularity of Twitter we collected some posts related with crime and criminology, in the state of São Paulo-SP Brazil. However, this data set is not a collection of crime reports. As the web language is characterized by diversity including flexibility, spontaneity and informality we need a classification rule to filter the documents which really are in the context. The proposed methodology works in a two step framework. In the first step we partition the text database into smaller data sets which define text collections with characteristics (not necessarily directly observable) which allow a better classification process. This enables the usage of parallel computing which decreases the time process required for the technique execution. Later on each subset of the data induces a distinct classification rule with a Supervised Machine Learning technique. For the sake of simplicity we work with *KMeans* and *KMedoids* and liner *SVM*. We will present our results in terms of speed and classification accuracy using various feature sets, including semantic codes. Analysis with distinct classifier induction techniques as *Random Forest*, *Logistic Regression*, and *Boosting* are also provided. An application with a huge data set of 1,600,000 tweets written in English proofs the method's efficiency.

Keywords: Text classification, machine learning, partition clustering, feature selection, edit distance.

1 Introduction

Text data originating from the web are a rich source of information. There is a huge amount of data, from many different users who report and complain about any subject of interest. In general, it is said that social web can give a good portrait of public opinion. Furthermore, it is easier to deal with text data rather than audio, images or videos.

Nevertheless, the usage of these texts requires specific analysis. The language itself is hard to study as a consequence of its subjective nature and the influence of many individual, economic, social and regional aspects, among others.

Text data is characterized by the fact that it includes unusual forms of usage like the usage of irony and metaphors. It must be added that an aggravating factor is that many words are polysemic, which means that they have more than one meaning. The word *bank* is a good example of word with polysemy.

Moreover, the web's flexibility, spontaneity and informality intensify the challenges in studies and analysis.

It might also be noticed that the web texts are getting smaller and smaller. Today, it is really common to share texts with status updates and which means that, unlike blog posts, the users usually write many times a day. These updates can be related to any subject; sport events, natural disasters, election campaigns or entertainment.

1.1 Preliminary Considerations

In this way, we can see that even for texts which share common words there can be texts in diverse topics. If the final aim is to use exclusively the texts in a given context we notice the need to apply a method to verify which texts do appertain to the intended context. Hence, we are facing a *Text Classification* or *Text Categorization* task, discussed in more details in Sub-section 1.1.1.

Pustejovsky and Stubbs [2012] describe *Text Categorization* briefly as the task to correctly sort a collection of elements into the proper category.

To simplify, let's say that the categories represent the phenomenon that we want to analyze and/or forecast. Then, there is a set of two classes ('to belong to the context' vs 'not to belong to the context') which enables us to refer to it as *binary* classification.

There are numerous works which make use of small texts in the study of a social phenomenon. Sakaki et al. [2010] propose a notification system for earthquake events which monitors Twitter posts and sends notifications instantly. Tumasjan et al. [2010] evaluate Twitter as a tool for predicting elections in Germany. Souza and Meira Jr [2016] use Twitter's data to detect infectious disease hot spots. In a hundred million (100,000,000) available posts the authors filter exclusively the posts which denotes the author's personal experience.

The references presented reinforce the need to study *Text Categorization*. The usage of textual information can be impaired by a bad classifier performance such that methods which minimize this undesired possibility should be strongly pursued.

As our data naturally comes in a digital form and as the data set can achieve high dimensionality we want an automated way of doing this task. Consequently, we want a classifier which learns the classification decision criterion automatically. This means that the *Text Classification* process is *machine learning-based*. More specifically, a *supervised learning* process.

1.1.1 Text Categorization

Figure 1, boxes (A), (B) and (D), represent an ideal procedure to classify a text collection. In box (A) there is initially an universe of texts, in which each color denotes a distinct subject. Let's say we want to collect the ones with a given topic, represented in pink/violet color. Filtering the collection by the presence or absence of a set of expressions we end up with a smaller collection although still hybrid. It includes the desired texts but also others (represented in grey). This collection of texts are the input to a classifier developed by algorithms in a computer, represented in box (B). The classifier outputs, for each text, the corresponding label which represents whether the document belongs to the subject of interest, shown in box (D).

Pustejovsky and Stubbs [2012] defines steps in a *Supervised Learning* framework, like *Text Classification* problems, briefly presented below:

1. To identify the target function. In the case of this work, the system should learn to identify if the *instances* are in the context of interest.

2. To choose a learning algorithm which derives a function able to distinguish the *instances* in the context of interest.

3. To evaluate the results according to a reasonable performance metric.

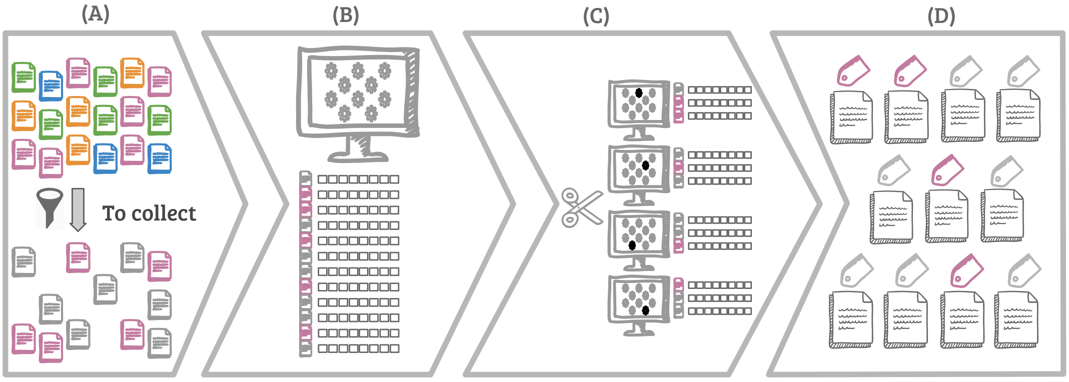


Figure 1: An overall representation from the intended procedure to classify a text collection is presented in (A), (B) and (D). Frame (D) describes the final aim; the classification task. The methodology that will be proposed in a two step framework is represented in (A)-(D). The intermediate step, in which the collection is broken into smaller subsets, is represented in (C).

1.1.2 Motivation

The classification task of a text collection crawled from web can easily achieve high dimensions. The speed and high quantity of data production and the availability of both its collection and storage enable one to work with much bigger data sets not easily seen until very recently.

Although the availability of this information sounds worthwhile it is well known that its usage is not trivial. The increase in data size is usually followed by an increase in computational effort. The extent to which the data set can be enlarged providing useful information to the classifier is hard to define. Meanwhile, the increasing complexity of the task can be assessed by means of time processing. For this purpose we conducted a simple experiment with *tweets*.

The *tweets* data set was collected during the year of 2013 in the state of São Paulo - BR with terms related to crime (label of interest: 'crime', label not of interest: 'anything else, different from crime'). More details regarding this data set can be found at Sub-section 5.2.

We selected samples whose sizes varied from 1000 (one thousand) *instances* to 15000 *instances* (fifteen thousand) in intervals of 1000.

The goals of the experiment are to observe both the classifier performance and the time needed to induce a classifier as a function of the training size data and distinct classifier induction methods.

As it is a *Supervised Learning* task, a concept detailed in Sub-section 2.1, each *instance* should be assigned to a *label*. Although we did not have the resources to manually classify this amount of data, we assigned each *instance* a label given by a classifier conceived through large experiments and described in dos Santos [2015]. It is important to observe that the proportions of 0.25 and 0.75, for 'crime' and 'non crime' labels, respectively, were kept in each scenario of the experiment. Figures 2 and 3 relate to this experiment.

For this experiment four classifier techniques were induced for each sub data set in the R statistical coding language [R Core Team, 2018]. The top frames describe linear classifiers, usually known to behave well in high dimensional spaces (concept in sub-section 2.1). On the other hand, the bottom frames present tree-based methods which can outperform the previous approaches if the relationship between the *features* and the response is highly non-linear and complex. The parameters used in this experiment were not evaluated with the usage of *cross validation* (presented in sub-section 2.4), they were obtained empirically.

The top frames describe linear SVM classifiers (soft margins) whose main ideas are presented in Sub-section 2.3. The top left plot refers to the implementation given by the `e1071` package [Dimitriadou et al., 2008] whilst the top right plot refers to the implementation given by the `LiblineaR` package [Fan et al., 2008]. Both linear SVM classifiers were induced with the cost parameter equal to 1 (one).

The bottom left frame describes a classifier induced with Logistic Regression Boosting with the `gbm` package [Ridgeway et al., 2006]. Briefly, this classifier can be described as a successive process of inducing a Logistic Regression model and fitting a decision tree to the residuals of the model. In this experiment we used 1000 (a thousand) trees, the number of splits in each tree as 1 (one) and the rate of which boosting learns as 0.01.

The bottom right frame plots the results for a classifier induced with a random forest using `randomForest` package [Liaw and Wiener, 2016]. This is a well known process to decorrelate the trees which implies also decorrelating the associate predictions. In this experiment we used 25 trees. For each tree only m of the original *features* are available, m being equal to the square number of the total number of different tokens in the data set.

Figure 2 shows F1 classifier performance metric (Y axis) as a function of the number of instances used in the training phase (X axis) by different classifiers (frames). The F1 metric is presented in Sub-section 2.5. With its inspection we can observe that, unless for the boosting approach, F1 is directly proportional to the number of instances used to train the classifier. With distinct increasing rates, the F1 grows as the data set size grows. This pattern is not observed for boosting maybe as a consequence of not specifying sufficiently the parameters.

Figure 3 plots the time needed to induce each classifier (Y axis) as a function of the number of *instances* (X axis) in the data set by different classifiers (frames). The plot clearly shows the fact that the bigger the training data set the more time consuming the classifier induction will be, unless there are some random variations.

Hence, the experiment shows that the induction cost grows up very fast as a consequence of the number of examples so that performance's increase is accompanied by a growth in computational effort.

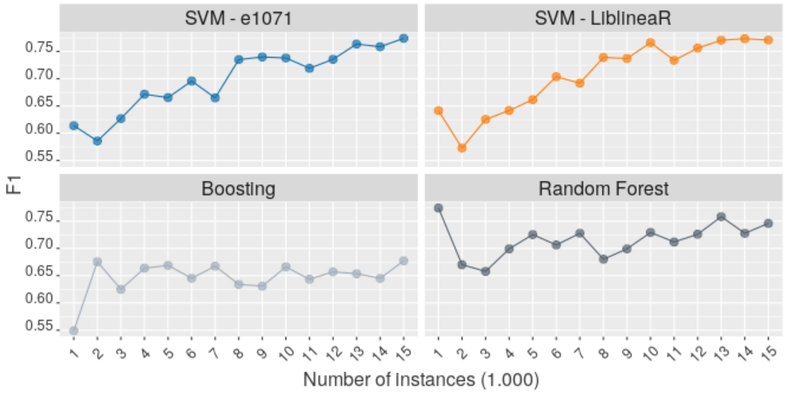


Figure 2: The classifier performance metric F1 as a function of the number of instances used by different classifiers (frames).

1.2 Main Objective

We aim to develop a methodology to enable the usage of large collections of text and possibly continuously increasing in a common desktop computer. Hence our goal is to perform an analysis in a simple computational structure.

It is reasonable to consider splitting the text data set into smaller subsets. In this way, the overall task is partitioned into smaller suitable tasks. Breaking the data set can be accompanied by a parallel stage which means the possibility of assigning each task to a distinct available core in the computer, which can decrease the global amount of time necessary for the task.

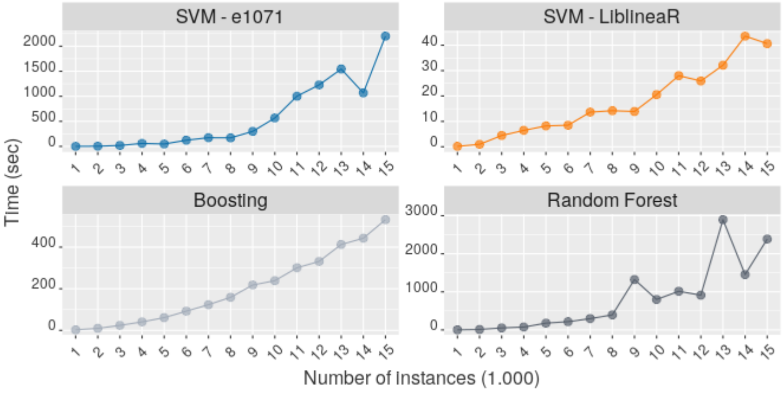


Figure 3: The time spent to induce a classifier as a function of the number of instances used by different classifiers (frames).

Figure 1, images (A)-(D), represents the overall procedure described in this work. To the overall *Text Classification* process indicated in (A), (B) and (D) it is included an intermediate step, represented in (C). Frame (C) corresponds to the split phase, occurring concomitantly in distinct cores of a simple computer. We might note, however, that frame (D) describes the final aim in the methodology, the classification task.

1.3 Work Organization

This work is organized as follows: Chapter 2 presents concepts and terminology with respect to steps (2) and (3) of *Supervised Learning* approach, cited in Subsection 1.1.1. Concepts related to text Annotation are in Chapter 3. The proposed methodology can be found in Chapter 4. The data set description and an extensive collection of experiments are presented in Chapter 5. Finally, the final remarks are listed in Chapter 6.

2 Fundamentals

This Chapter aims to present the concepts, terminology and more descriptions of the *Classification task*, *Machine Learning*, text representation, *feature selection*, text annotation, some classifier induction methods, cross validation and performance metrics to assess the classification quality.

2.1 Text Categorization, Text Representation and Machine Learning concepts

Throughout this work, texts are the individual units observed. They are referred as *documents* or *instances*. We use *Corpus* (or *Corpora*) for a collection of *documents*. The *words* are the measurable properties which characterize the observed units. In other words, *words* are the *features* and a *word* will be denoted as a *term* or *token*. Figure 4 (left side) presents these concepts.

The *documents* are represented in a common vector space and, consequently, the representation is called *Vector Space Model*. In this representation the order of the terms in a *document* does not matter so it can be referred as a *bag of words*. The sentence 'The hare is quicker than the tortoise' is equivalent to 'The tortoise is quicker than the hare' under this model. Jurafsky [2000] reinforces the strength of the assumptions and limitations of the *bag of words* representation claiming that the meaning of a document resides solely in the set of words it contains. Usually we observe many *documents* and, as a consequence, many distinct *tokens*. Hence, it is said that we deal with problems at *high dimension* and *sparse vectors*.

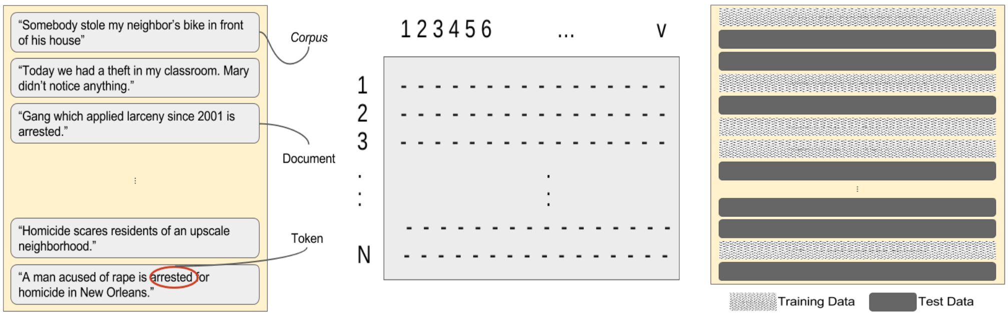
The *bag of words* representation starts by removing some extremely common words, known as *stop words*. This is a list of high-frequency words, considered to have little semantic weight. This characteristic is arguable since sentences as 'to be or not to be' or 'let it be' provide a counter proof. The aim of their elimination is to save space. *Stop word* removal can be considered a way of pruning highly frequent terms. Schütze et al. [2008] affirm, however, that in the Information Retrieval field, some modern systems do not use *stop word* lists. Based on this fact, the present work did not exclude *stop words* neither.

The *bag of words* representation includes assigning weights to the terms in each document. There are three most common weighting schemes: *term frequency - TF*, *inverse document frequency - IDF* and *term frequency - inverse document frequency - TF-IDF*, all described in Jurafsky [2000] and not detailed in this text.

Figure 4 (center) exemplifies these concepts. The data is presented in a matrix in which each line denotes a *document* and each column denotes one of the possible observed *tokens* in the *Corpus*. The weights, prioritizing the *tokens* most characteristic of a document, are represented as the entries of the matrix.

The derivation of the function described in the second step of the *Text Categorization* problem (presented in Sub-section 1.1.1) requires a labeled set of input-output pairs (the *supervised* approach) and a collection of *examples*. It means that an expert reads a set of initial documents and classifies its contents with some possible labels. This labeled data set is split into two distinct data sets, called *training* and *test* data sets. The *training* set is used to induce the classifiers and the *test* set is used to assess the quality of the results obtained, described in step (3).

The *test* set is not used in the induction process. We forecast the observations in the *test* set with the classifier developed with the *training* set. The classification rule's power to forecast is called the classifier generalization ability and is a desired quality. It is assessed and quantified by different metrics, depending on which characteristic it is essential to save. Sub-section 2.5 presents some of them. Figure 4 (right side) illustrates the *training* and *test* data sets.



"Somebody stole my neighbor's bike in front of his house"

"Today we had a theft in my classroom. Mary didn't notice anything."

"Gang which applied larceny since 2001 is arrested."

"Homicide scares residents of an upscale neighborhood."

"A man acused of rape is arrested for homicide in New Orleans."

Figure 4: Left: Typical expressions of a text classification task. Center: The *Vector Space Model representation*. Right: manually labeled data set splitted in *training* and *test* data sets.

2.2 Feature Selection

Short texts, whether explicitly constrained in size or not, crawled from the Internet are used by users with different profiles and purposes. Aggarwal and Reddy [2013] add that contracted forms of words and slang are prevalent in short texts. By its nature, the data sets correspond to usually informally written documents, suffering from grammatical mistakes, misspelling and improper punctuation. As a consequence, we can say that the observed language has great variability.

From this, it is reasonable to say that not all *features* are applicable both when grouping texts (intermediate step) and classifying them (main objective). In such texts, irrelevant *features* are referred as *noise features*.

In the *unsupervised* grouping phase, *noise features* imply concentration effects in the grouping methodology, a concept presented in Sub-section 4.1.

In text classification tasks, adding *noise features* to the document representation increases the classification error on new data, which is an undesirable behavior.

Thereafter, it is fundamental to use *feature selection* techniques defined by Pustejovsky and Stubbs [2012] as the process of finding which *features* in your data set are most helpful in solving the desired task. *Feature selection* concerns the choice of useful variables to characterize the elements in a study and by so it can be also thought of as a text representation step.

In addition, *feature selection* is particularly relevant for classifiers which are expensive to train, like the ones with a huge number of *features* as in the experiment presented in Sub-section 1.1.2.

Besides this, it can be anticipated that the grouping phase will use distance-based methods. Aggarwal and Reddy [2013] observe that, in the presence of many *noise features*, pairwise distances between data points can become similar and incapable of revealing which are the similar *instances*.

The step of representing the texts with meaningful *features* is an essential step. It is an attempt to represent the *documents*. Consequently, it is the first step in the analysis which means that it is relevant to the quality of the overall task. Some reasons to target the text representation step with meaningful *features* are listed below.

- part of the *noise* language is removed,
- the usual high dimension is 23 considerably decreased,
- it results in significant features which are going to play a meaningful role in the subsequent steps,
- it reveals some (if it exists) hidden pattern structure comprehensible and helpful to the subsequent algorithms.

Aggarwal and Reddy [2013] cite the existence of *Feature Selection* and *Feature Extraction*. Both of them aim to lower computational complexity, to decrease required data storage and to provide better data interpretation. Nevertheless, in the former there is a transformation from the original space into a smaller space whilst in the latter the aim is to select a smaller subset of *features* maximizing relevance to the target.

Among the *Feature Extraction* methods, a famous one is called Principal Component Analysis (PCA) which has previously been applied in the data set of this work with results described in dos Santos [2015]. Since the new space projection with lower dimension implies loss of interpretation, it is preferable to work with *Feature Selection*.

In this work, three main *feature selection* approaches will be applied. Sub-sections 2.2.1, 2.2.2 and 2.2.3 describe distinct ways of using the observed vocabulary. Sub-section 2.2.4, in turn, presents an approach to enrich the observed data with external information, a task described as *text annotation*.

2.2.1 Pruning observed vocabulary/Pre-processing the text

As described in Sub-section 2.1, the *Vector Space Model* with *Bag of Words* representation is applied in this work. To reduce the dimension and the possibility that the classifier uses *noise*, it is common to clean the text in a process called pre-processing. In other words, we clean the *documents* removing undesirable characters and *tokens*. The pre-processing step depends on the data set as well as the domain of analysis. Consequently, it will be described in details in Chapter 5. It can be anticipated, though, that there is the removal of *tokens* with frequency lower than five (our threshold). The Term-Frequency weighting scheme was used since in previous experiments, documented in dos Santos [2015], the weighting scheme did not produce decisive differences in the results.

2.2.2 Most common *tokens*

Even after removing rare *tokens*, the *feature* space can still be considered large. A common approach in authorship attribution, called Frequency-based Feature Selection [Schütze et al., 2008], is to extract the most frequent words in the *Corpus*. As described by Stamatatos [2009], this approach needs the definition of the number of most frequent *tokens* which are going to be considered as *features*. In accordance with Stamatatos [2006], we tried an approach with the 1000 (thousand) and 2000 (two thousand) most common *tokens* observed.

As said previously, this work did not exclude the *stop-words*. As a consequence, the most frequent *tokens* refer to closed-class expressions (prepositions, articles, etc) followed by open-class expressions (nouns, verbs, adverbs, adjectives, etc).

The numbers 1000 and 2000 of most common *tokens* are attempts to define the vocabulary dimension which represents adequately the vocabulary variability. In other words, the numbers can be chosen by the user in accordance with the domain of study, the language specificity or the machine-learning technique to be applied.

2.2.3 Information Gain metric

Beyond the methods already presented, there is also the possibility of applying *feature selection* measures. Since it has a low computational cost we decided to apply a metric in the class of Filter methods known by Information Gain or Mutual Information.

The metric computes, for each class c in the dependent variable, a utility measure $A(t, c)$ for each *token* t and it selects the k *tokens* that have the highest utility measure values [Schütze et al., 2008]. For the Information Gain metric the utility measure corresponds to the dependence between the class and the *feature*. Equation (1) shows how this dependence is described mathematically. It relates the joint occurrence of a given feature f and a specific class c (joint probability in the numerator) with its overall occurrence (marginal probability in the denominator). The probabilities are estimated by relative frequencies.

If knowing about the occurrence of feature f does not provide any information about class c, the probabilities described in Equation (1) will be similar. This implies that the fraction approximates 1 and consequently the metric approximates 0 (zero). Therefore, *features* with measures close to 0 are considered less informative to class c.

$$I(X = c, Y = f) = \log_2 \frac{P(X = c, Y = f)}{P(X = c)P(Y = f)} \tag{1}$$

As well as the most common *tokens* approach, we decided to select the 1000 (thousand) and 2000 (two thousand) most relevant *features* based on the Information Gain metric.

2.2.4 Text annotation

Besides the vocabulary, the usage of the most frequent terms and the usage of terms selected by the Information Gain measure, we also tried to represent the texts with another approach called text annotation. More specifically, we tried *part-of-speech tags*, commonly called *POS* tags and *semantic annotation*.

In a Machine Learning framework it is known that it is not enough to provide the computer with a large amount of data. The data needs to be prepared in such a way that the machine can find patterns in it and the algorithm used is capable of inducing classifiers.

The *Corpus* annotation task is the process of assigning a marker or label to each word in a *Corpus*. It is an attempt to add interpretative information to the data. In other words, the *Corpus* annotation effort aims to add value to the collection of texts by summarizing other information which can reveal some relevant aspect to subsequent algorithms. Although the use of annotated data does not guarantee improvement in the algorithm results, it is a common practice in text data methodologies.

The work of Pak and Paroubek [2010] employs *part-of-speech tags* to help describe text sentiments. The presence (or absence) of *features* such as nouns, verbs, pronouns, adjectives and adverbs describes the texts as predominantly positive, negative or with neutral emotion. The authors describe satisfactory results and they cite and exemplify plausible patterns found.

It is interesting to observe that some of the patterns can be considered expected like the more common use of personal pronouns in subjective texts. However, they also observe superlative adjectives being used to express emotions and opinions whereas comparative adjectives were mostly employed to state facts and to inform, occurring predominantly in objective documents. That is, the *POS* tags reveal known facts but also highlight aspects which would not be noticed without their use.

Based on these results, it is natural to desire these annotations to be applied to data bases for small texts. They can possibly reveal useful hidden patterns for the next steps in the methodology but they definitely imply *feature* space reduction. Though this, later clarified in Chapter 3, consider two simple examples. With the usage of morpho-syntatic annotation the *tokens eyes* and *mouth* would be both annotated as *Noun*, whilst with the usage of semantic annotation let's say they would be annotated as *parts of the body*. In other words, two distinct *tokens*, which are typically represented with two distinct dimensions in the *feature* space, are mapped to the same dimension if using the annotation schemes. That way, the *feature* space is reduced.

Since the data of this work is naturally digital, automatic Text Annotation was chosen. The Text Annotation process is usually studied and developed in the field of Natural Language Processing (referred as NLP). Given the fact that there are many concepts associated with it, we present the related methodology in Chapter 3.

We might point out, however, that applying automated text annotation has known drawbacks. The first difficulty is the need to access a robust and accurate NLP tool, an aspect which we consider, for this work, solved. There is also the fact that, as the classifier which assigns the labels is usually based on random variables, it will produce *noise features* due to unavoidable errors made by the parser [Stamatatos, 2006]. Last, we point that annotation *features* are relatively computationally expensive, an idea of this computational effort is presented in Sub-section 5.2

2.2.5 Comments on Text Representation

The way to represent each *instance* in a Machine Learning-based approach is decisive to the classifier performance. The representations detailed in this text describe the ones used in the experiments. Beyond them, it is possible to apply any other way of selecting *features*. Work such as Guyon and Elisseeff [2003] is a guideline to choosing appropriate methods based on specific aspects of the problem being analyzed.

With the aim of overcoming the limitation of the *bag of words* model disregarding word-order, Stamatatos [2009] suggests the usage of *word n-grams* (or word sequences). Nevertheless, the authors mention that the high and undesirable cost of the increase in dimensions and sparsity do not necessarily lead to accuracy growth. The authors also cite the possibility of working with n-grams at the character level which, according to them, would better suit *noisy* text data, such as those crawled by the web.

2.3 Learning Algorithm - Support Vector Machines

The *Machine Learning* literature has a variety of methods for *binary classification*. Among them, a particular simple class of models work well for text data: the linear classifiers. These functions are defined by James et al. [2013] as two-class classifiers that decide the label of a given instance based on which side of a separating hyperplane it lies.

A Linear Classifier and its generalization deserve special attention [Boser et al., 1992]. Their concepts and detailed description can be found in James et al. [2013] and are not presented here. Despite the terminology abuse, the Linear Classifier with soft margins is referred in this text as *SVM*, a shortening for *Support Vector Machine*.

In this approach, there is only one parameter to be tuned. We refer to it as C, from cost, and it works as a regularization parameter, a parameter to penalize the number of instances inside the margins of the classifier.

Figure 5 exemplifies some concepts. The two labels of interest are represented by circles and crosses. The classifier is the hyper plane most distant from the closest opposite *instances*. The two dashed parallel hyper planes represent the *margins* of the classifier. The parameter C controls the number of instances which lie on the wrong side of the margin.

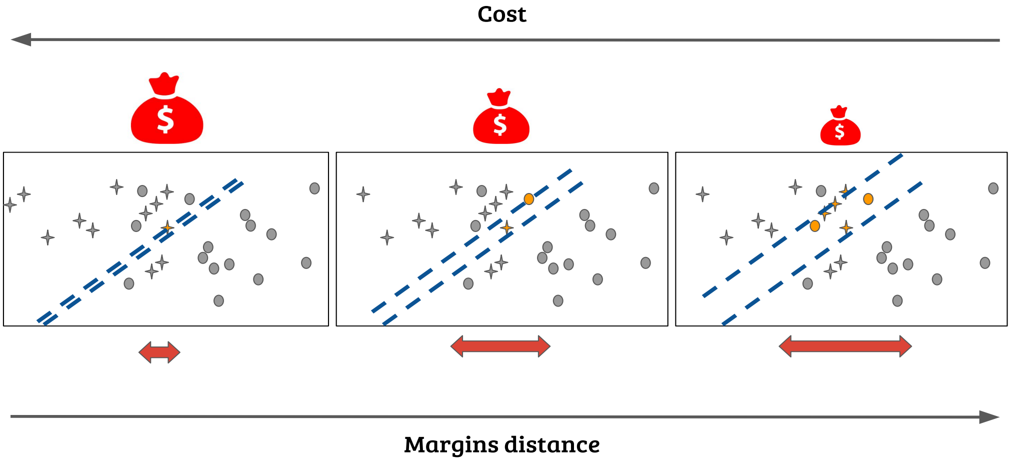


Figure 5: Linear Classifier with soft margins as a function of the regularization parameter C.

C parameter works by controlling the so called *bias-variance trade off* [Schütze et al., 2008]. *Bias* and *variance* are two competitive concepts widely discussed in James et al. [2013]. *Bias* refers to the error of modelling a real life problem with a model, a simplification. *Variance* denotes the extent to which a classifier output would change using a different training data set.

The *bias-variance trade off* explains our preference for a linear classifier in text categorization tasks. Text applications usually have high-dimensional spaces and linear models in those spaces are quite powerful [Schütze et al., 2008]. In the same space, nonlinear models have more parameters to fit and they are more likely to learn from *noise*. It ends up training the data too precisely which prevents it from generalizing well; an undesirable behavior referred as *overfitting*.

It is important to pay attention to the fact that despite this consideration there have been previous attempts to use non-linear SVMs (with *kernel tricks*). The substantial increase in training complexity did not improve the classification performance. These results can be found in dos Santos [2015].

It must be pointed out that, on the other hand, SVM has a linear training time. This mitigates against the usage of large training data sets, a well known fact cited by Schütze et al. [2008] and confirmed by the experiment described in Sub-section 1.1.2.

2.4 Cross validation

In a machine learning-based approach the main objective is to induce a classifier with good predictive power when classifying unseen observations. As it has been cited previously, in machine learning terminology, we want a classifier with great generalization power. Therefore, it is necessary to assess how well a given classifier can be expected to perform on independent data [James et al., 2013]. As common practice, this is done by estimating the error rate in a non-previously observed text data set.

Considering the computational cost and the potential correlation between sample results, we chose to work with *k-fold cross-validation*. More specifically, the experiments included *10-fold cross-validation* since $k = 10$ yields test error rate estimates that do not suffer from high bias nor from high variance [James et al., 2013].

2.5 Performance Metrics

Assessing the quality of the classifier induced is essential to understand the limitations of the function obtained or to highlight aspects of the training data set.

As the tasks approached in this work are binary, the most common way of doing this is by obtaining and analyzing a Confusion Matrix. This is the name given to a 2x2 contingency table relating the labels given in the labeling phase and by the classifier for the test set *instances*. An example can be found in Table 2.5.0.1.

Table 2.5.0.1: Confusion Matrix for a binary classifier.

Technique/System	Manual Classification		Total
	Label of Interest	Label not of Interest	
Label of Interest	a	b	a+b
Label not of Interest	c	d	c+d
Total	a+c	b+d	n

The entries a-d in this table are usually named as *true +*, *false +*, *false -* and *true -*, respectively.

It is natural to think about *accuracy*, the fraction of items correctly classified, given by the relation $\frac{a+d}{n}$, as a measure of classification quality. However, as the data sets are usually unbalanced with respect to the classifier labels, using the *accuracy* measure may incur efficiency misinterpretation. In an extreme case, if the system only assigns labels to the dominant class the system might seem to be efficient as the overall percentage of correctly classified *instances* is high. However, this implies a large number of mistakes for the opposite label.

As a consequence, it is common to use two measures which are based on the *true +* cases: *precision*, given by $\frac{a}{a+b}$, and *recall*, defined as $\frac{a}{a+c}$. It is easy to see that both measures takes values in the $[0, 1]$ interval. We can notice as well that high values of *precision* imply low values for *false +*, whereas high values of *recall* imply low values for *false -*. *Precision* and *recall* can also be expressed as functions of *specificity* and *sensitivity*, concepts of Diagnosis tests.

It is always possible to have a classifier with *recall* 1, it just needs to assign all instances to the *Label of Interest*. This practice implies an expressive value of *false +*, which decreases the *precision* and we clearly see that there is a trade-off situation.

In an attempt to simplify even more the quality analysis in this work we will use the *F1* measure. This is the harmonic mean of *precision* and *recall*, with same weights for both measures. The harmonic mean acts in a way to penalize a classifier with too different values for the metrics. In other words, for a set of classifiers, the one with the biggest *F1* value is considered to be the best, with a good balance between the two values.

Figure 6 and Table 2.5.0.2 help exemplifying how the metric is obtained considering the splitting scheme adopted. The image describes a collection of *test* texts, documents manually classified which were not used in the classifier induction step.

They are divided in eight groups and each group is assigned to an available core with its respective induced classifier.

All the *test instances* are classified and each smaller sub data set has a Confusion Matrix associated, represented in Table 2.5.0.2. The table also describes the time needed by each classifier to predict the *test* labels. The Classification task, however, must be assessed taking into account the overall performance of the classifier. An ensemble matrix corresponds to the sum of the entries, shown in the bottom line of the table and identified as Total. The *F1* metric is obtained based on this ensemble matrix. As the tasks were executed concomitantly the time associated with the ensemble matrix is the longest prediction time observed.

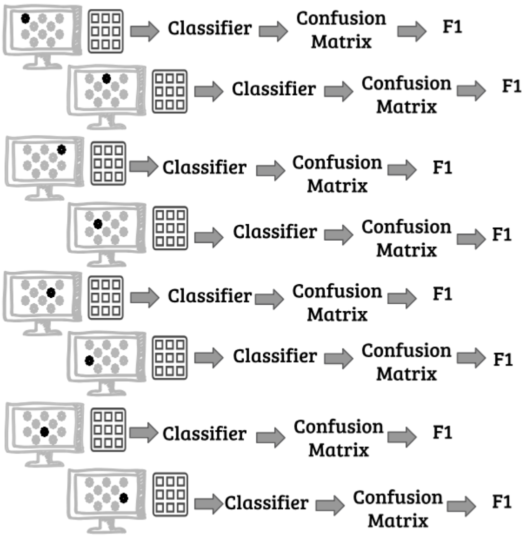


Figure 6: Example of a confusion matrix and an F1 matrix for 08 groups.

	Confusion Matrix				Time
Group	Lab.:NLb Clas.:NLb	Lab.:NLb Clas.:Lb	Lab.:Lb Clas.:NLb	Lab.:Lb Clas.:Lb	Prediction
1	109	17	31	77	1.134
2	97	2	8	6	1.150
3	207	12	24	30	1.125
4	435	14	37	33	1.190
5	249	13	22	17	1.120
6	240	51	36	222	1.134
7	139	24	33	45	1.109
8	254	18	37	22	1.139
Total	1745	136	239	441	1.190

Table 2.5.0.2: Example of a combined Confusion Matrix with 8 *clusters*.

3 Morpho-syntatic and Semantic Text Annotation

Section 2.2.4 explains the motivation to use text annotation in this work. As mentioned there, text annotation usage is desirable in text classification methodology because it enables one to swap a huge *feature* set (all the observed vocabulary) by many fewer *features*.

One common way of annotating texts is by labeling them with respect to morpho-syntatic categories. This task is called *part-of-speech* tagging, cited in this text by the acronym *POS*.

However, semantic coherence is not usually used as a definition criterion for *parts-of-speech* [Jurafsky, 2000]. Consequently, when using *POS* tags, the notion of meaning is not being directly explored. Considering this, another approach was derived trying to use semantic annotation.

As well as the morpho-syntactic tags, this semantic approach would suit the methodology. Different words mapping to the same semantic code highlight that two distinct lexical forms have closely related meanings (like t-shirt and trousers) which implies the already cited desired *feature* space reduction.

3.1 Automatic annotation

Although the *Corpus* is not huge we considered it unfeasible to annotate it manually. In addition to our lack of a capable professional to perform the task, in our methodology we thought it desirable not to depend on human capabilities. In the contexts in which it is expected to apply our methodology, the database is expected to grow fast making it unfeasible to require a step in the process which would be time-consuming and possibly financially costly.

According to these facts we chose to use automatic annotation. As a consequence, the endeavour of annotating thousands of Portuguese texts required the use of tool(s). Given our resources and time availability we chose to work with Linguakit [Cillenis, 2017a] for the POS tagging and the UCREL USAS system [UCREL, 2018b] for the semantic tagging.

Although the number of Natural Language Processing (NLP) tools available has grown considerably in the last decades finding tools for this work was not trivial. As the clear priority, a tool which could work with Portuguese language was needed. Besides this, we needed the tool to be free and not restrict the total number of annotations done daily. In future, the methodology might be able to accommodate a greater volume of documents.

In Sub-section 3.2 we briefly describe the free tools used in this work. The concepts and methods associated with the annotation processes adopted are presented in Sub-sections 3.3 and 3.4.

3.2 Linguakit and USAS-UCREL

Gamallo and Garcia [2017] describe Linguakit as a multilingual suite for NLP tasks, written in Perl and developed by Cillenis [2018a], an enterprise devoted to developing technology for language products with strong knowledge and academic background.

The University Centre for Computer Corpus Research on Language (UCREL) [UCREL, 2018a] is a Research Centre of Lancaster University which develops NLP methodologies based on *Corpora*. The group focuses on practical outcomes and modern foreign languages (such as Portuguese), among other topics. The group has been more than forty years in existence and is considered innovative in *corpus* construction and annotation.

Both websites, Cillenis [2017b] and UCREL [2018c], provide any interested user with a broad range of NLP tools. Their advantage is that they can be accessed online or by downloading the source code and executing it by command line or accessing their server. The user specifies the language, the task desired and the directory with the files to be processed as described by the documentation. Also, both websites have no limits in the number of access to their tools daily, which is something desirable once the amount of documents to be classified is large.

3.3 Syntactic annotation

Syntax is the study of formal relationships between words [Jurafsky, 2000]. In our overall methodology our final aim is to distinguish between labels of interest. To employ syntactic concepts is an attempt to explore the way words depend on other words in a sentence with the perspective that these dependency relations help in differentiating the labels of study.

As stated by Jurafsky [2000], *POS* tagging is the process of assigning a lexical class marker to each word in a *Corpus*. Table 3.3.0.1 describes the output for an example document from the Linguakit annotation tool.

For each document, the analysis starts with the tokenization phase which means the process of segmenting the text in meaningful units. For short texts this step is not really significant as many of them have only one clause.

Later, each word is associated with its *lemma* which means its canonical form. The *lemma* for the word in English '*go*' represents the forms: *go*, *goes*, *going*, *went* and *gone*. The word '*ir*' in Portuguese represents the forms: *vamos*, *vou*, *iria*, *irei*, *fui* and *foi*.

Afterwards, there is the step of assigning a morpho-syntatic tag (from a set of predefined tags) to each *token*. The process can be rule-based or stochastic.

Word	Lemma	Annotation	Code
ele	ele	Pron. Pes. 3aPes. Masc. Sing. Indef.	NCMS000
pára	parar	Verbo Princ. Indicat. Pres. 3aPes. Sing.	VMIP3S0
todos os dias	todos os dias	Adv. Geral	RG00000
para	para	Prepos. Simples	SP00000
ir	ir	Verbo Principal Infinit.	VMN0000
para	para	Prepos. Simples	SP00000
a	o	Determ. Artigo Fem. Sing. Indefin.	DAIFS00
faculdade	faculdade	Nome Comum Fem. Sing.	NCFS00
.	.	Pontuação	Fd

Table 3.3.0.1: Example document with *POS* annotations given by Linguakit software.

In this tagging process, the algorithm takes as input the desired *Corpus* to be annotated and a specified tagset and a set of finite possible syntax classes to give the *tokens*.

The tagset used by Linguakit has 193 elements as a consequence of the complex verbal conjugation and nominal inflection system of the Portuguese language [Garcia and Gamallo, 2015].

Initially, the classifier uses 21 tags for disambiguating the morpho-syntatic category (noun, verb, adjective, adverb, preposition, pronoun, etc.) of each word. Then, the information related to gender, number, tense, etc is taken from a labeled dictionary. The Linguakit morpho-syntatic annotation process works as a mixture of a stochastic (first step) and a rule-based (second step) approach.

The stochastic approach means a supervised task in which the technique derives the probabilities of a given word having a given tag in a given context based on a training *corpus*. It usually applies the framework of picking the most-likely tag for a given word [Jurafsky, 2000].

Usually, this task is solved using a Hidden Markov Model (HMM) framework.

Jurafsky [2000] describes the usage of the assumption that the probability of a word depends only on its tag and the assumption that the tag history can be approximated by some finite integer number (usually two) of recent tags. Jurafsky [2000] claims the estimates are obtained by maximum likelihood and that HMM taggers work by searching for the most probable sequence of codes.

Garcia and Gamallo [2015] describe a similar approach to the Linguakit POS tagger, despite a different dependency structure of the model for computational efficiency. Garcia and Gamallo [2015] handle the task with a simpler treatment considering it as an individual classification problem.

The annotation codes (sequence of 7 characters) are available at Cillenis [2018b] and show the complex verbal conjugation and nominal inflection of the Portuguese language.

3.4 Semantic annotation

Semantics is the study of the meaning of linguistic utterances [Jurafsky, 2000]. As well as using syntax, our final aim is to explore word meanings with the aim that they are going to help the main classification task.

As stated by UCREL [2018d], the semantic tags work as sets of words related at some level of generality with the same mental concept.

The Semantic tagger available online is a framework which has been researched and designed since 1990. It is an automatic multilingual semantic annotation system which means that it employs a unified semantic classification scheme [Piao et al., 2016]. It works for twelve languages, including Portuguese.

The list of tags, known as the *tagset*, used was based on a reference work described in Piao et al. [2016]. The revised *tagset* results in 21 major discourse fields expanding into 232 category labels. This is presented in Table 3.4.0.1. The groups include not only synonyms and antonyms but also hypernyms and hyponyms and a set of auxiliary codes. UCREL [2018d] documents the sets, exemplifying them with some terms. For instance, the word *cheerful* would be coded as E4.1 and the word *frustrated* would be coded as *E4.2*, both words being considered emotions.

(A) general, abstract terms	(B) the body, the individual	(C) arts, crafts
(E) emotion	(F) food, farming	(G) government and public
(H) architecture, housing, the home	(I) money, commerce in industry	(K) entertainment, sports, games
(L) life, living things	(M) movem., locat., travel transp.	(N) numbers, measurement
(O) substances, objects, materials, equip.	(P) education	(Q) language, communication
(S) social actions, states, processes	(T) time	(W) world, environment
(X) psychol. actions, states, processes	(Y) science, technology	(Z) names, grammar

Table 3.4.0.1: Major discourse fields in the revised semantic tagset used in USAS.

The USAS UCREL project is intended to be a large-scale high-quality multilingual lexical resource. Its main idea is to automatically translate the core English semantic lexicon using bilingual dictionaries and other available bilingual lexicons. In other words, the aim of the project is to extend an existing English semantic annotation tool to cover more languages with the distinction of employing a lexicographically-informed semantic classification scheme [Piao et al., 2016].

The idea was to develop semantic lexicons for other languages by translating English expressions to the target languages and later transferring the semantic tags. Some corrections were applied later, if possible. The Portuguese lexicon was largely generated automatically with single-word and multi word expression entries.

An extended discussion of the multilingual *Corpus*, bilingual lexicons used and experimental results to estimate the quality and lexical coverage are presented in Piao et al.. Despite the fact that some results indicate that the lexicon needs to be expanded, the USAS tool was chosen to be used in this project being the only one tool known for semantic tagging.

Table 3.4.0.2 presents the output of the system for an observed text.

Token PT	Lemma	POS	Code
Uma	um	DI0	N6- T1.1.1 Z5
vez	vez	NCFS	N6- T1.1.1 Z5
eu	eu	PP1	Z8mf
ia	ir	VMI	Z99
ser	ser	VMN	A3+ Z5
roubado	roubar	VMP	G2.1-
em	em	SPS	A5.1+ G2.2+ A1.1.1
a	o	DA0	Z5
saida	saida	NCMS	Z99
de	de	SPS	Z5
um	um	DI0	N1 T3 T1.2
show	sho	NC	MS
que	que	PR0	Z99
fui	ir	VMI	Z99
em	em	SPS	A5.1+ G2.2+ A1.1.1
a	o	DA0	Z5
capital	capital	NCFS	I1.1 M7/G1.1 Q1.2
e	e	CC	Z5
quando	quando	RG	Z5
fui	ir	VMI	Z99
ve	ve	VMN	Z99
o	o	DA0	Z5
ladrao	ladrao	NCMS	Z99
estudava	estudar	VMI	P1 X2.4
cmg	cmg	NCMS	Z99

Table 3.4.0.2: Example of Semantic annotated sentence.

4 Proposed Methodology

This project aims to develop a classification task framework for huge amount of small texts in usual desktop computers. Ideally, the classifier will have predictive power but not be time consuming as described in Sub-section 1.2.

In the context of this work, a good classification scenario is one which reveals interesting patterns in the mining activity inducing a classifier with good predictive power. In addiction, it is desirable to enable usual desktop users to use text data without the need of a specific or advanced computer. In other words, we want a framework applicable to regular home computers. Figure 7 illustrates the proposed methodology.

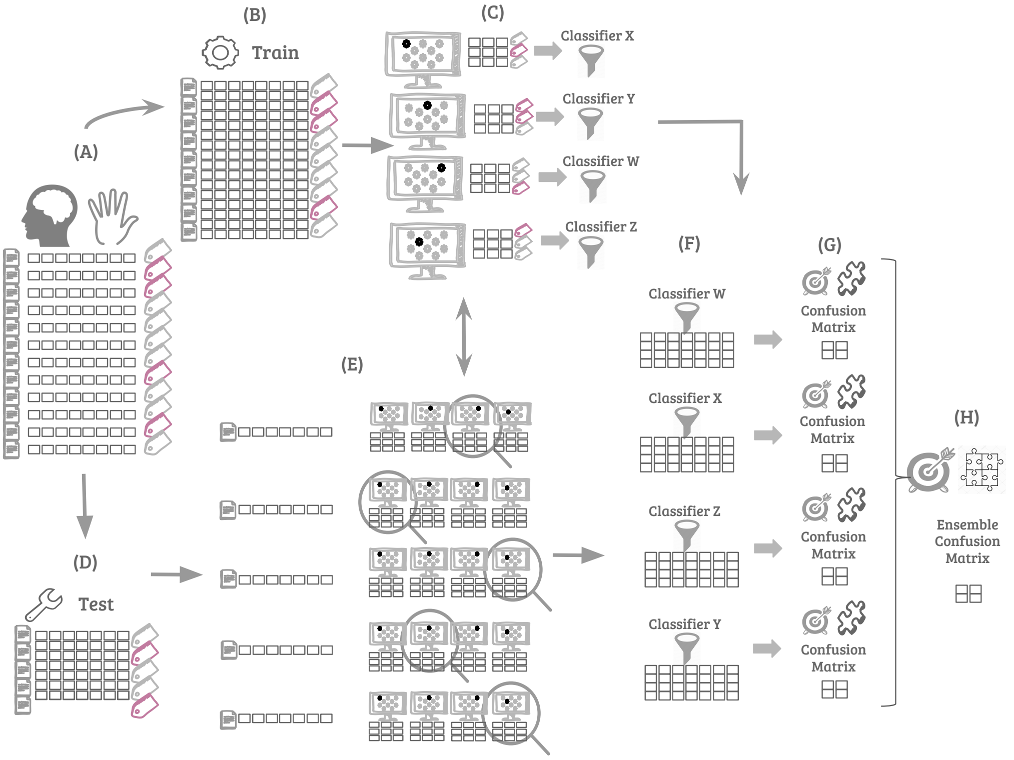


Figure 7: Scheme describing the *training* process and *test* process.

Since the proposed methodology is a machine learning-based approach it starts with a manual labelling phase described in frame (A). Each document is represented by any convenient text representation and it is accompanied by a label (pink denotes the label of interest and gray expresses the label not of interest). The labels were assigned, one by one, by some capable human, usually with expertise in the topic of study. Hence, at this point, there is a labeled set of input-output pairs, a collection of *examples*.

As cited in Sub-section 2.2.4, the work of Pak and Paroubek [2010] employs annotated data to determine text sentiments. Just as designed by Reddy and Mohammed [2013], this work aims to apply Syntax and Semantics with the purpose of identifying useful patterns, probably not easily perceived by human analysts, that co-occur in the form of groups. Although the usage of annotation implies *feature* space reduction, the *bag of words* model still presents the inconvenient of being high-dimensional, sparse and producing a loss of order in the text representation. As a consequence, this methodology aims to work with another type of text representation. Instead of regarding the *instances* in a structured data-set (data frame with fixed number of columns for all rows), the texts will be represented as sequences of annotated Syntax or Semantic codes. By sequences we mean that the texts correspond to a single character which concatenates the annotation codes for each *token*. Sequences correspond to a relatively new type of data representation which is capable of maintaining the order of its constituents.

The manually labeled data set serves two purposes: to allow the classifier induction and to assess the classifier performance. The former is represented in the top part of the image and it is named the *training* data set (frame (B)). The latter is being represented at the bottom part, being called the *test* data set (frame (D)).

The classifier induction process is both computationally and time consuming. With the aim of decreasing the computational effort and to reduce the time needed to accomplish the task of inducing a classifier based on the whole collection of human effort available, it is desirable to split the *training* data set. At the frame (C), the whole collection of texts is split into smaller subsets, balanced with respect to the number of instances. This is the stage described as the *clustering* process. It is a fundamental step in the methodology and it might comprise sequences as data structure. Despite this sophistication, this phase should not be effort or time consuming. After all, the *clustering* process works as an intermediate step in the whole analysis. The class of *clustering* techniques which are distance-based accommodate any type of data set, given the existence of a meaningful distance metric. Thanks to researches in fields as Genetics, similarity measurements between characters are available and they will be incorporated in this work. Consequently, we apply a technique in this class to handle sequence data type.

Notice that at the frame (C) each subset is assigned to a distinct core in the computer. It is clear that each core and each subset specifies its own classifier. The final aim of the methodology is to classify the texts, the step in frame (C). The split phase occurs from frame (B) to frame (C), describing the so-called step two framework of this methodology.

The process is then followed by the classifier performance assessment. It is a step which evaluates or quantifies how well the classifier would behave in the presence of unseen data.

Each non-previously observed *instance* is assigned to one of the computer's cores and its respective classifier (induced in the training phase). This part is represented in frame (E). In this way, each non-previously observed *instance* is assigned to the classifier (and core, consequently) whose *instances* are more similar to it. This similarity concept is represented in the magnifying frame. The indication of which group is most similar to a given *instance* can be done by several ways depending on aspects such as the shapes of the *clusters* defined or the amount of computational effort. We emphasize the methodology flexibility once it allows each classifier to be distinct. For instance, consider a scenario with 04 classifiers. The approach enables the classification task to have two linear classifiers and two non-linear classifiers, each of them defined by a distinct, specific set of parameters.

Each *instance* is classified by its respective classifier, represented in frame (F) and finally it is possible to determine the texts in the context of interest according to the classifiers.

Hence, at this stage, each test data set *instance* has two labels: the one given manually (frame (A)) and the one resulting from the classifier (frame (F)). To assess the classification process quality the labels are compared. The Confusion Matrices which make explicit the joint frequencies are represented in frame (G). There is one Confusion Matrix for each core being used for the analysis. In order to have an overall perception of the classification quality, the Confusion Matrices are joined together in an Ensemble Confusion Matrix, represented in frame (H).

Comments on sequences are presented in Sub-section 4.1. Next, Sub-section 4.2 describes aspects of the intermediate step, the grouping phase. This includes some related work, and theoretical aspects. Metrics to quantify similarities between characters are discussed in Sub-section 4.3. Sub-section 4.4 is dedicated to describe how each test instance or any non-previously observed instance is assigned to the available classifiers and computer cores.

4.1 Texts as Sequences

Like DNA, RNA, mRNA, polypeptides and proteins, texts have a linear structure and can be represented as sequences. The advantage of representing texts as sequences include taking into account the order of the elements in a text and the possible usage of methodologies which are impaired by high-dimensional and/or sparse data structures. Usually, the elements order in a text is considered by using immediate neighbors and the distance between elements is a parameter. Bigrams and trigrams refers to the usage of neighbors of size one and two respectively. Hence, the approach discussed in this text corresponds to a new approach, which differentiates from the usual ones representing each text by its order dependence.

When using some *clustering* algorithms, like distance-based, bi-clustering, graph-theoretic or Markov based *grouping* methods, with high-dimensional data, it is possible to incur in a known undesirable effect called distance-concentration effect. Zimek [2013] points out that this effect means that far and close neighbors have similar distances which reduces the utility of the measure for discrimination. Hence, representations which avoid this undesirable effect are needed for this work. That way, sequence representation, beyond ensuring the order and neighborhood maintenance between the constituents of a given text, prevents negative implications in the *clustering* method.

As an example, consider the *document* described in Table 3.3.0.1. Its associated sequence, which corresponds to the concatenation of the annotation codes assigned to the *tokens* would be:

NCMS000VMIP3S0RG00000SP00000VMN0000SP00000DAIFS00NCFS00Fd

4.2 *Clustering*

As described in Section 1.2, a fundamental step of this work concerns grouping texts. In the context of this work, the groups are created to facilitate a data mining main task, working as an intermediate step.

The *grouping* text phase aims to make the classifier induction task easier computationally. This phase denotes the task of defining sets of items in a way that the *instances* are similar with respect to the other *instances* in the same group and the latter are distinct between them.

4.2.1 Related work

The decisive work of Blei et al. [2003] came up with a new paradigm in text *clustering*. With more than 26,000 citations up to June 2018 it opened a new way of grouping texts together, modeling the topics by probability distributions with simple way of inferring the parameters.

In the same context of huge data bases, Du et al. [2015] approaches the problem using non-parametric Bayesian tools. They develop rule induction by the text content with a dynamic time evolving component using a time point process. The idea is to enable the number of clusters to accommodate the increasing complexity of online text content. The method is adapted for use in continuous time.

Once the approach presented in this text is mainly thought to work for small documents we might call attention to the fact that documents constrained in length increase the challenge of *clustering*. Subsets of documents can be related by topic but written with completely distinct vocabulary from each other. As a consequence, short documents do not usually share terms which makes the task even more difficult.

More recently, Yin and Wang [2016] adapted Blei's model for short texts and the possibility that the topics would automatically change with the time. Its algorithm, despite its sophistication, requires low computational time to be applied. On the other hand, David C Anastasiu and Karypis [2013] believe short *documents* might be enlarged or complemented with external data, in a process called knowledge infusion. In this case, external documents, are accessed and the original documents structure is enlarged with their contexts to help the *clustering* process.

Both previous references group the texts by topic. The methodology presented in this work does not group by topic.

Clustering phase is an intermediate step in the analysis and consequently there is no interest in the context being revealed by the groups. It is considered that the documents have one out of two possible topics ('to be in the context of interest' or 'not to be in the context of interest') which will be given by the classifier in the final step.

Following Reddy and Mohammed [2013], this methodology aims to identify some patterns that co-occur in the form of *groups*. Nevertheless, the patterns do not need to be directly observed. After all, the sequence patterns are not going to be analyzed. They just need to work well for the next step of the methodology. We aim to define similar texts in the groups, but they are similar syntactically or semantically, not similar as topics, improving the classifier performance in the final step.

Clustering instances in this methodology aims to divide the overall task in smaller tasks. The idea is to split a huge task/effort in smaller tasks. Each smaller task corresponds to a group of similar documents which, as a consequence of this similarity, induce a better classifier in the following/final step. This approach is known as divide and conquer. It suits perfectly the proposed methodology since it splits a huge effort, as a classifier induction for large amount of instances, in smaller tasks. In addition to being a consolidated approach it depends on simple desktop cores availability, something which increases with the natural computers structure progress.

4.2.2 Methods

Alelyani and Huan [2013] define the *clustering* task as the unsupervised classification of instances into groups. Although there is an extensive literature devoted to grouping elements, this methodology focuses on a classical approach, the *partition* cluster method. *Partition* algorithms address the task as an optimization problem; minimizing a function. Given a number k of clusters, the methods are based on the concept of a representative (also called prototype) such that each group has one representative. Each prototype is chosen to represent the general pattern observed in its respective *cluster*. Each *instance* is assigned to the cluster which the representative is most similar with. As the assignments occur, the representative changes (its characteristics are updated) and the process is repeated for all *instances*, a specified number of times or until some threshold is achieved. Thus, there is an iterative process in which there are adjustments of the *cluster* membership for each data point until some convergence criterion is met.

As pointed out by Bouras and Tsogkas [2012], *partition clustering* methods are known for their low complexity, in comparison with other *clustering* methods. As a consequence, *partition* clustering methods suit large *document* databases, just like the ones this methodology is applied to.

Its most popular variant is called *K-Means* [Lloyd, 1982] in which the representatives are called *centroids*. One *centroid* is a virtual instance in which each *feature* corresponds to the average (mean) of the *features* of the other observations in the same *cluster*. Figure 8 (A), exemplifies these concepts. The groups are colored with distinct colors and the crossed circles denote the representative of each group. Notice that the representatives correspond to the center between the observations, easily seen in a two-dimensional representation. In an analogy with Mechanical Physics, the *centroids* are the center of mass of their respective clusters.

Also with the virtual representative concept, there is the *K-Medians* algorithm. It differs from *K-Means* by taking the median along each dimension to create the group prototype. Thus *K-Medians* are recommended for non-numeric data or when in the presence of *outliers*.

Finally, *K-Medoids* also has an alternative way of assigning as a representative one of the elements in the *cluster*. More specifically, this is the element which minimizes the overall distance to the other elements of the group. In other words, the representative is not a virtual concept but an observed *instance*. Figure 8 (B) clarifies this. In this plot, in opposition to the frame (A), the crossed circles are under a given observation. The representatives, in this case, work as leaders of each group. They are real observed *instances* which will represent each group. Just like *K-Medians*, *K-Medoids* are recommended to non-numeric data or when in the presence of *outliers* and its computational cost is higher than the other approaches. *K-Medoids* is the most appropriate *clustering* technique for this methodology because it enables the usage of a *partition-based clustering* method for a data structure which does not have a clear concept of median, or a central tendency metric once sequences are nominal realizations.

The drawbacks of *Partition* algorithms include the influence the initial representatives have in the overall analysis. Aggarwal and Reddy [2013] also add that *Partition* algorithms are good at capturing groups with spherical shapes, a characteristic difficult to observe directly by users. Nevertheless, the most often mentioned disadvantage is the need to specify the number k of groups.

It is convenient to observe, though, that the specification of the number of *clusters* is not an inconvenient in this methodology. The use of *cluster* techniques aims to group the *documents* but there is no need for meaningful partitions, something desirable in most *cluster* applications. This aspect reinforces the auxiliary role of *clustering* in the analysis. The number k of groups is a quantity specified by the user expressing how many available cores can be employed in the whole task. It does not work as another parameter to be tuned in the analysis.

4.3 Levenshtein metric of similarity

The possibility of using a distance-based *clustering method* as a *partition* algorithm is due to the fact that those algorithms just rely on the specification of a similarity

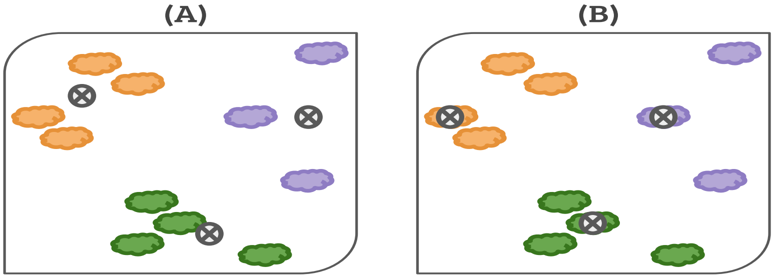


Figure 8: Representation of *K-Means* and *K-Medoids* representatives.

metric matrix. This means that it is possible to apply the *clustering* methods to any data structure as long as there is a meaningful metric to represent similarity between the *instances*.

Some of the possible metrics to be applied can be found in Gabadinho et al. [2011] and will not be presented in this text. The choice of a metric to quantify the similarity between two sequences of annotated codes took into account practical aspects. We chose to use the Levenshtein metric, also referred as Optimal Matching or Edit Distance.

The Levenshtein metric for two sequences x and y corresponds to the minimal cost of transforming one sequence into the other with allowable edition operations. This operations are the insertion, deletion, substitution of characters and the shift of parts of the sequence. Levenshtein metric has the advantage of working for sequences of distinct lengths which is necessary when comparing texts. And last but definitely not least, Levenshtein metric counts with a well known implementation available. Sub-section 5.6 describes how to access the implementation applied in the experiments for this study.

4.4 Non-previously observed *instances* assignment

A decisive step in this methodology concerns how to assign a non-previously observed *instance* to a given classifier. In other words, how to decide to which computer core with its respective classifier should an unprecedented *instance* be assigned.

Based on concepts of Hierarchical *clustering*, discussed in Reddy and Bhanukiran [2013], this work tried three different approaches, which are represented visually in Figure 9.

One possible approach is named in this text as *Medoid-based*. It assigns a new *instance* to the *cluster* whose representative, the medoid, is most similar with. It is an analogous to the centroid-based agglomerative method of *clustering* in the Hierarchical literature. Its drawback is the fact that representing a group by its prototype is more suitable for *clusters* with a spherical or ellipsoidal shape, a characteristic difficult to be observed directly by any user. However, it is the simplest possible approach since it requires the derivation of a fixed number of distances depending on how many groups are being considered in the analysis. It corresponds to the frame (A) in Figure 9.

Instead of considering exclusively the prototype, a second approach tries to assess the overall similarity between a given unprecedented *instance* and the group. In this approach, the distance between a new *instance* and each of the *instances* on a given *cluster* is obtained. The distances are averaged and the new observation is assigned to the group with smallest mean distance. This overall behavior is expensive to compute , however. In this text, this approach is named as the SmallestMeanDist being inspired by the Group Averaged assignment in the Hierarchical *clustering* literature.

As a third possible option this study also evaluated the impact of assigning a non-previously observed *instance* i to the group which the most similar member to i belongs to. It does not consider the overall structure of the *cluster*. We can also say it corresponds to a 1 nearest neighbor concept. This approach is referred in this text as KNN and the Hierarchical *clustering* as Single link. It is capable of grouping nonelliptical shaped groups although it is sensitive to *noise* and time consuming as it is based on the notion of ordering distances.

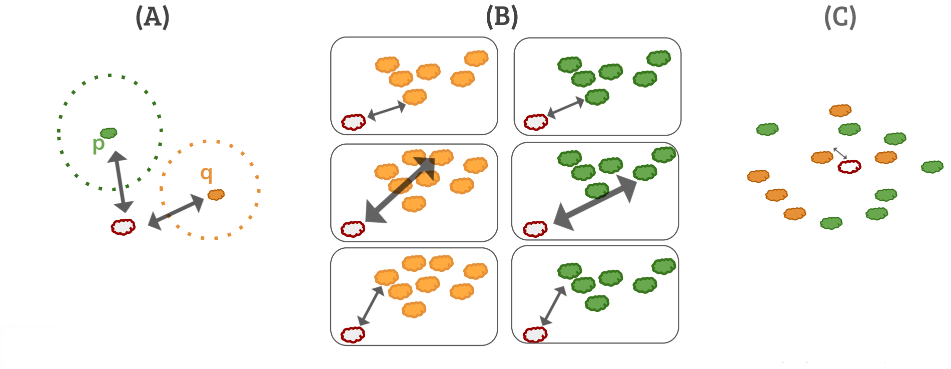


Figure 9: The three distinct ways of assigning a non-previously observed instance. (A) By the representative/prototype. (B) Considering the group with smallest mean distance. (C) Assigning to the same groups as the most similar instance in the training data set, called the nearest neighbor.

5 Experiments

The methodology proposed was evaluated and tested with some experiments. The goal was to investigate it's advantages, applicability, difficulties and limitations.

There is an initial experiment in sub-section 5.1 followed by sections devoted to technical aspects. Details about data set are presented in sub-section 5.2. Parameters taken into account like feature selection process, number of groups evaluated, cross validation schemes and performance metrics adopted are presented in sub-sections 5.3, 5.4 and 5.5, respectively. Technical aspects related to R programming language are presented in sub-section 5.6.

In sub-section 5.7 we present a big experiment overview and their results, in sub-section 5.8.

Following, with the aim of investigating the methodology behaviour when presented to other types of classifiers, we present a second experiment in sub-section 5.9.

At last, sub-section 5.10 presents results of the proposed methodology applied to a different context and language.

5.1 Initial Experiment

As a first attempt, we explored the proposed approach in a Brazilian data-set with thousands of documents related to criminology published in Twitter social web. This data base is fully described following, in sub-section 5.2.

For this first approach of grouping texts, we used the well known partition algorithm K-Means (sub-section 4.2.2).

Hence, we represented the texts in VSM format (sub-section 2.1) with absolute counts/ frequencies as weights.

Later, each cluster defined a separate data set which is applied to a classifier induction process - Support Vector Machine (SVM) methodology (sub-section 2.3). For all scenarios in the study we adopted a cross validation of size 10.

An inspection of the clusters obtained revealed that the groups defined by K-Means algorithm on VSM representation defines extremely unbalanced groups, independently of the number of clusters adopted. Figure 10 shows this. It corresponds to the percentage of instances assigned to the biggest cluster defined by K-Means (axis Y) for different number of groups (axis X) (detailed following, in sub-section). Each dot represents one part in the cross validation scheme.

For all groups (axis-x) we can observe variability in the relative size of the biggest group defined (axis-y). We see lines of dots in the image in the whole extension of axis-x. It means straightforward that K-Means ends up defining unbalanced groups with a single group with big amount of observations whilst the others were assigned really small amount of instances.

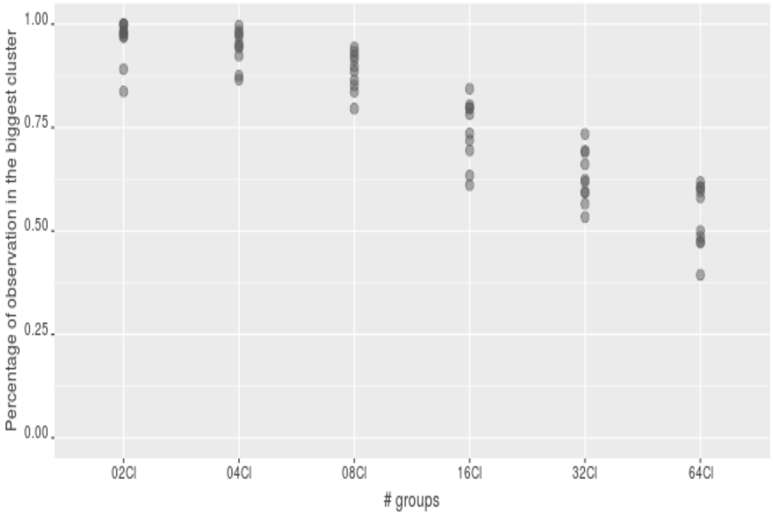


Figure 10: Percentage of instances assigned to the biggest cluster defined by K-Means with vocabulary in VSM text representation.

The non-supervised step was conceived with the aim of decreasing computational effort in classifiers induction process for big data sets. Hence, a result in which data gets unbalanced is not desirable. After all, to accumulate an expressive amount of instances marginally decreases the task and, consequently, it's computational time.

As a consequence of this result, we thought about other ways of representing the text. Being aware of the role of text representation in grouping them, we did the same experiment varying ways of representing the instances. Our main idea was to explore this aspect, in an experiment controlling by text representation effect.

Thus, we replicated the experiment selecting *features*. We got the 1000 and 2000 most frequent terms, an approach referred here as *topTerms* (sub-section 2.2.2). Also, we worked with 1000 and 2000 terms selected by Information Gain feature selection metric (sub-section 2.2.3) and referred here as *infoGain*.

The respective Figures 11 and 12 show results similar to those described previously.

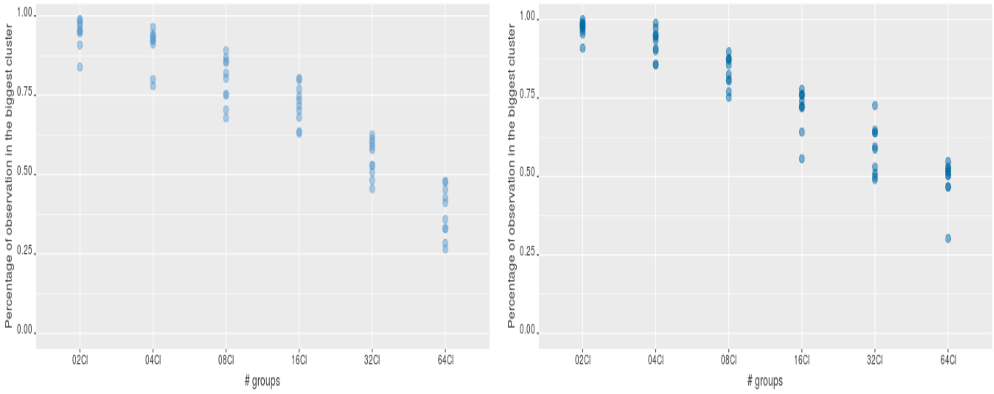


Figure 11: Percentage of instances assigned to the biggest cluster defined by K-Means with: top 1000 terms (left) and top 2000 terms (right).

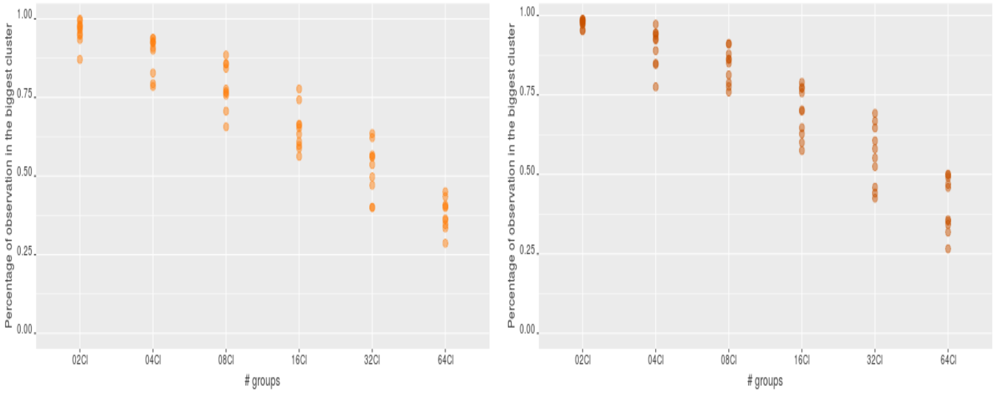


Figure 12: Percentage of instances assigned to the biggest cluster defined by K-Means with: 1000 terms (left) and top 2000 terms (right) selected by Information gain metric.

Hence, this initial experiment shows the need of other ways of grouping texts. Let us be reminded that it must be simple and effective in defining balanced groups focusing the main effort to the classifier induction process.

To discover the best way of grouping texts we designed experiments. Section 5.7 describes what aspects were taken into account. Before this necessary overview, we discuss remaining characteristics regarded to the experiments.

5.2 Data sets

To enable the experiments, we used a previously known and labelled data set. It was conceived as a collection of data from social network Twitter whose posts are called tweets henceforth in this text. Since the tremendous variability of topics discussed in this social web, we decided to focus our attention in the tweets related with crime and criminology. It was hand labelled for *crime* and *não crime*. Thus, this initial collection was used as train and test *corpus*/data sets enabling the classifiers induction as well as its performance assessment.

5.2.1 Area

This initial *corpus* was collected in September 2013 for tweets written in Portuguese and which were posted in the area of São Paulo state, in Brazil.

5.2.2 Terms

We collected tweets which had at least one of the tokens in a given set. This set was defined as the most popular expressions denoting crimes. It corresponds to variations in number (singular or plural) and gender (male and female) of 25 root expressions and it is presented in Table 7.0.0.1, in Appendix. As a simplified version, we present the nominal forms of the initial roots listed in Table 5.2.2.1.

Agressão	Atentado violento ao pudor	Furto	Pedofilia	Saída de banco *
Arrastão	Clonagem	Gangue	Prisão	Sequestro
Arrombamento	Estelionato	Golpe	Quadrilha	Tráfico
Assalto	Estupro	Ladrão	Refém	Violência doméstica
Assassinato o	Fraude	Morte	Roubo	Vítima

Table 5.2.2.1: Nominal form of the keywords used in Twitter collection. The colors represent a general type of crime, in Portuguese. Blue: *contra a pessoa*, Green: *contra o patrimônio*, Red: *sexuais* and Purple: general expressions.

5.3 Feature selection

Based on the considerations set out in Section 2, we decided not to use *stop word* lists. In contrast, we remove terms which appear less than five times in the whole collection. Thereafter, we prune infrequent terms.

5.4 Number of Groups

To explore different possible users' profiles or infra structure availability context, the scenarios were observed for some defined number of groups. The range was chosen to be 02, 04, 08, 16, 32 and 64. Figure 13 gives a visual insight of this range.

The lower number of groups represents the users with availability of a single computer, with 02 or 04 cores available to accomplish the desired task. On the other hand, users with 32 or 64 cores available are those with much more infra-structure such as someone with access to servers and/or laboratories. The intermediate number of *clusters*, 08 and 16, describe scenarios where the users have access to more than one single simple computer or a machine with slightly advanced settings.

It might be noted that, as the industry develops, the number of available cores increases. Thus, it is reasonable to suppose that the scenarios with higher number of clusters will be possible in the near future, even for simple users.

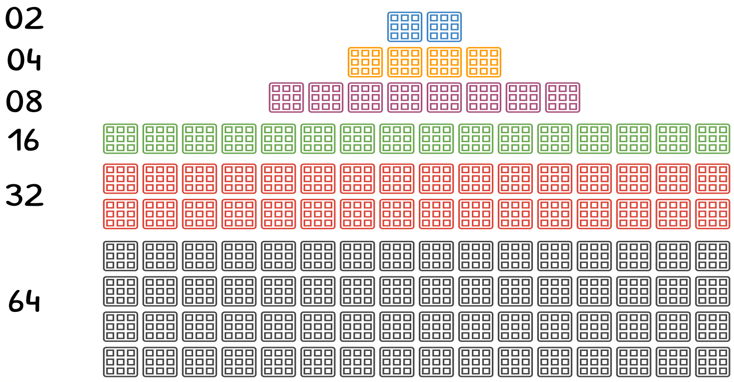


Figure 13: The distinct number of *clusters* to be analyzed. The range is intended to cover the distinct user profiles or infra-structure available.

5.5 Cross validation and Performance Metrics

To assess the classifiers behavior with unseen data we used Cross Validation (subsection 2.4) of size 10.

The performance metric used in this work as well as the detailed description of its adaption to the context of this work was presented in sub-section 2.5.

5.6 R language

All the analysis were done at R statistical software (R Core Team [2018]). The R language is popular between data analysts, as a consequence of its flexibility dealing with different types of data and methodologies and the quality of its presentation and visualization.

We remember that the perspective of this work is to enable common users to classify big amounts of data in usual computers. In view of this, R language emerges as a natural programming language to use in the experiments. It seems to mimic the typical environment a common user of this study would work.

This work used distinct packages which are presented following, grouped by functionality.

- to read JSON data:
  RJSONIO: Temple Lang and Wallace [2018];

- to treat text data easily:
  tidytext: Silge and Robinson [2016];
  stringr: Wickham [2018];
  tm: Feinerer and Hornik [2018];

- to manipulate data:
  dplyr: Wickham et al. [2018];
  tidyr: Wickham and Henry [2018];

- to enable parallel programming:
  foreach: Microsoft and Weston [2017];
  iterators: Analytics and Weston [2018];
  doParallel: Corporation and Weston [2017];

- to do beautiful plots:
  ggplot2: Wickham [2016];
  ggthemes: Arnold [2018];
  jcolors: Huling [2018]

5.7 Overview

In general, the proposed methodology of this work can be thought as a combination of three main parts which are dependent of each other and we address them as steps.

The first, second and third steps are called *text representation*, *Clustering* (or *Grouping*) and *Classifier* steps, respectively.

The *text representation* step corresponds to the choice of which information represents each instance in the second phase. *Clustering* step aims to differentiate the best manner to group similar instances, both in the training and in the test phase. At last, *Classifier* step corresponds to the classifier induction which takes into consideration the classifier induction method as well as features choice to represent the instances.

Scenarios were conceived with the main aim of exploring the methodology behavior with respect to different combination of possibilities in the described steps. Table 5.7.4.1 describes the differences between each scenario and it helps clarifying experiments' details.

5.7.1 Text Representation

We say there are five distinct ways of representing the instances to the first phase: *vocabulary*, *top terms*, *information gain*, using *Syntax annotation* and using *Semantic annotation.*

By *vocabulary* we mean representing each text (instance) by its respective terms, an usual approach. Although a common way of representing the texts and despite the pruning of non-frequent terms, the *vocabulary* approach implies a high dimension representation and, consequently, a sparse representation. More specifically, in this case, the texts are represented by the presence (or not) of the terms in a collection of 4749 terms. Then it is said to have complete information with the cost of dealing with objects of high cost computations. Scenarios with *vocabulary* text representation are described in lines 1 and 2 in Table 5.7.4.1.

Thereafter, we thought about investigating other vocabulary approaches with lower dimensions. It is also a common practice to work with the vocabulary by focusing on the most frequent terms, an approach described in sub-section 2.2.2. This approach was addressed as *top terms* approach. In this case, there is the choice of how many terms consider. Scenarios with *top terms* text representation are described in lines 3 to 6 in Table 5.7.4.1.

Also, we tried to reduce the vocabulary selecting terms by an appropriate metric. More precisely, we adopted Information Gain metric, described in subsection 2.2.3 and accordingly this approach is called *Information Gain.* Scenarios with *Information Gain* text representation are described in lines 7 to 10 in Table 5.7.4.1.

For *top terms* and *information gain* approaches we varied the number of tokens in two; one near half the total vocabulary (rounded to 2000) and another one with approximately a quarter the total vocabulary (rounded to 1000).

There are also the possibility of using text annotation as documents characteristics. We explored both *syntax annotation* and *semantic annotation*. In these cases there are considerably dimension reduction. Using *syntax annotation* - scenarios in lines 11 to 14 in Table 5.7.4.1 - and *semantic annotation* - scenarios in lines 15 to 18 in Table 5.7.4.1 - each text is represented by 191 and 920 annotation codes, respectively.

Text annotation enables expressive dimension reduction. Also it is subject to automated system errors, an aspect not present in other text representations. Henceforth, text annotation brings the possibility of information loss, which can compromise the grouping results. Based on this, besides the text representation in Vector Space Model, we also considered text representation in Sequence format, as described in Section 4.1.

We might keep in mind some facts. In our context, it is reasonable to think of a syntactic or semantic labels sequence. Syntactic and semantic codes correspond to well defined and well known sets. Obviously, we might be aware that, specifically for semantic codes, there can be subjectivity in its definition. Nevertheless, they correspond to functions which can map multiple terms to the same code, providing dimension reduction. Consequently, these annotation schemes can perhaps reveal a non directly observable pattern which are defined much easier than pattern in sequences made of many possible constituents, as words in a language. Text annotation scenarios using sequence representation are described in lines 12 to 14 and 16 to 18 in Table 5.7.4.1.

5.7.2 Clustering

Clustering phase corresponds to the step devoted to grouping similar elements. It is decisive in decreasing the computational time spent in the whole task. As a consequence, it is thought to be an aid in the overall classifier induction processes. This means that, the chosen approaches were conceived as simple and popular solutions. We tried three distinct approaches depending on the text representation.

In the case of VSM representation data corresponds to counts, in other words, to numbers. Hence, in this specific case of text representation, we use the well known *K-Means* (section 4.2.2) algorithm whose implementation used in this work uses Euclidean distance. Lines 2, 4, 6, 8 and 10 of Table 5.7.4.1 correspond to it's scenarios.

However, sequences correspond to non-numeric information units. They work as levels of a category variable. Thus, it is not possible to think of grouping them taking a centroid, which is typically defined as an element whose features are the average of the member's group features. Then, sequences are grouped with *K-Medoid* method (section 4.2.2), an analogous of *K-Means* oriented to elements of non-numeric observations. *K-Medoid* implementation in this work uses Levenshtein distance, presented in Section 4.3 . Lines 12 to 14 and 16 to 18 of Table 5.7.4.1 correspond to it's scenarios.

Moreover, we considered a third approach thought to be simpler. It works as a *vanilla* version serving as a baseline of comparison. The *Random Split* clustering method corresponds to randomly assigning instances to a previously defined number of groups. It does not use any specific intelligence or distance metric. It works as the simplest version of *divide and conquer* approach. Lines 1, 3, 7, 11 and 15 of Table 5.7.4.1 correspond to it's scenarios.

Another aspect to be taken into account corresponds to the way non- previously observed instances are grouped in the defined *clusters* at the train phase.

In the case of *K-Means* approach it is straightforward. New instances are assigned to the groups which their representatives are more similar with. In Table 5.7.4.1, the test instances assignments are *centroid* and *medoid* to *K-Means*, respectively.

Also, for *Random Split* grouping approach, new instances are easily assigned to the groups: it occurs randomly. These cases are represented in lines 1, 3, 7 and 11 of Table 5.7.4.1.

But for sequences, and *K-Medoid* consequently, there can be distinct ways of assigning new instances to previously defined groups. It can occur assigning a new instance to the group whose representative is more similar with - cases represented in scenarios 12 and 16. Also, it can be assigned to the group whose average distance to all its constituents is the smallest (lines 13 and 17). A third possible approach is to assign a new instance to the group which has the most similar instance, an approach called *1NN* and represented in scenarios 14 and 18.

5.7.3 Classifier Induction

The final step is also the most important one. At this point, the main aim is to train a classifier. It corresponds to represent each instance with meaningful features. Once most popular classification techniques take data in columnar format we evaluated scenarios with this characteristic; which corresponds to VSM representation.

Also, we know *terms* correspond to the most representative features unit, specially after some pre-process step as we did. Therefore, we induced a classifier representing each text by its original units: it's terms.

As *syntax* and *semantic* annotation also represent texts in columnar format they were evaluated in scenarios described in lines 11 and 15.

5.7.4 Scenarios overview

Observing 1st, 2nd and 3rd steps together we observe each scenario role in understanding the methodology. Once we know usual classifiers need columnar format data the third step is basically the same in all cases.

In Table 5.7.4.1 we can think of two main text representations schemes, VSM (from lines 1 to 10) and sequences (from lines 11 to 18). Instances represented in VSM format can be clustered with *K-Means* algorithm while instances taken as sequences are grouped by *K-Medoid* algorithm.

Lines 1 to 10 correspond to ways of defining useful features. Lines 11 to 18 correspond to distinct ways of addressing non-observed instances to the groups in scenarios with sequence text annotation representation.

	1st step: Text Representation			2nd step: Clustering		3rd step: Classifier height
	Representation	Scheme	Dimension	Grouping	Test instances assignments	Features
1	Vocabulary	Vector Space Model	4749	Random Split	Random	Terms
2	Vocabulary	Vector Space Model	4749	KMeans	Centroid	Terms
3	Top Terms	Vector Space Model	1000	Random Split	Random	Terms
4	Top Terms	Vector Space Model	1000	KMeans	Centroid	Terms
5	Top Terms	Vector Space Model	2000	Random Split	Random	Terms
6	Top Terms	Vector Space Model	2000	KMeans	Centroid	Terms
7	Information Gain	Vector Space Model	1000	Random Split	Random	Terms
8	Information Gain	Vector Space Model	1000	KMeans	Centroid	Terms
9	Information Gain	Vector Space Model	2000	Random Split	Random	Terms
10	Information Gain	Vector Space Model	2000	KMeans	Centroid	Terms
11	Syntax Annotation	Vector Space Model	191	Random Split	Random	Syntax Annotation
12	Syntax Annotation	Sequence	24119	KMedoid	Medoid	Terms
13	Syntax Annotation	Sequence	24119	KMedoid	Lowest Mean Distance	Terms
14	Syntax Annotation	Sequence	24119	KMedoid	1NN	Terms
15	Semantic Annotation	Vector Space Model	920	Random Split	Random	Semantic Annotation
16	Semantic Annotation	Sequence	23781	KMedoid	Medoid	Terms
17	Semantic Annotation	Sequence	23781	KMedoid	Lowest Mean Distance	Terms
18	Semantic Annotation	Sequence	23781	KMedoid	1NN	Terms

Table 5.7.4.1: Experiments scenarios discriminated by steps.

5.8 Results

We made experiments correspondent to each line in Table 5.7.4.1. Once K-means approach was proved to be non-interest as discussed in Sub-section 5.1 experiments described in lines 02, 04, 06, 08 and 10 are not considered and they will not be presented.

To represent the behaviour of VSM text representation methods we select the scenario associated with best results for scenarios in lines 01, 03, 05, 07, 09, 11 and 15 in the referred table. With the goal of being brief and also to allow a better visualization we eliminate from the following discussion results related to scenarios in lines 03, 07. In other words, scenarios which were conceived with a deep feature space reduction resulting in around 1000 (a thousand) final features. As we expected, this extreme feature reduction meant a dramatic decrease in the classification algorithm learning power and accordingly in its predictive power.

As described previously, for each number of groups, scenarios 01, 05, 09, 11 and 15 were evaluated considering F1 quality fit metric and experiments are replicated in a cross validation scheme of size 10. Results of this experiments are presented in Figure 14 which plots observed F1 in *box plots*, which are a concise way of visualizing empirical distributions. Each frame is for a specific number of groups.

We explored SVM also varying cost parameter between values: 0.1, 1, 10, 100, 1000. And once more with the aim of being brief, we present results for linear SVM with costs 10 and 100 only once they correspond to the best results. In the following plots, bright colours are for cost 10 and dark tones are for cost 100.

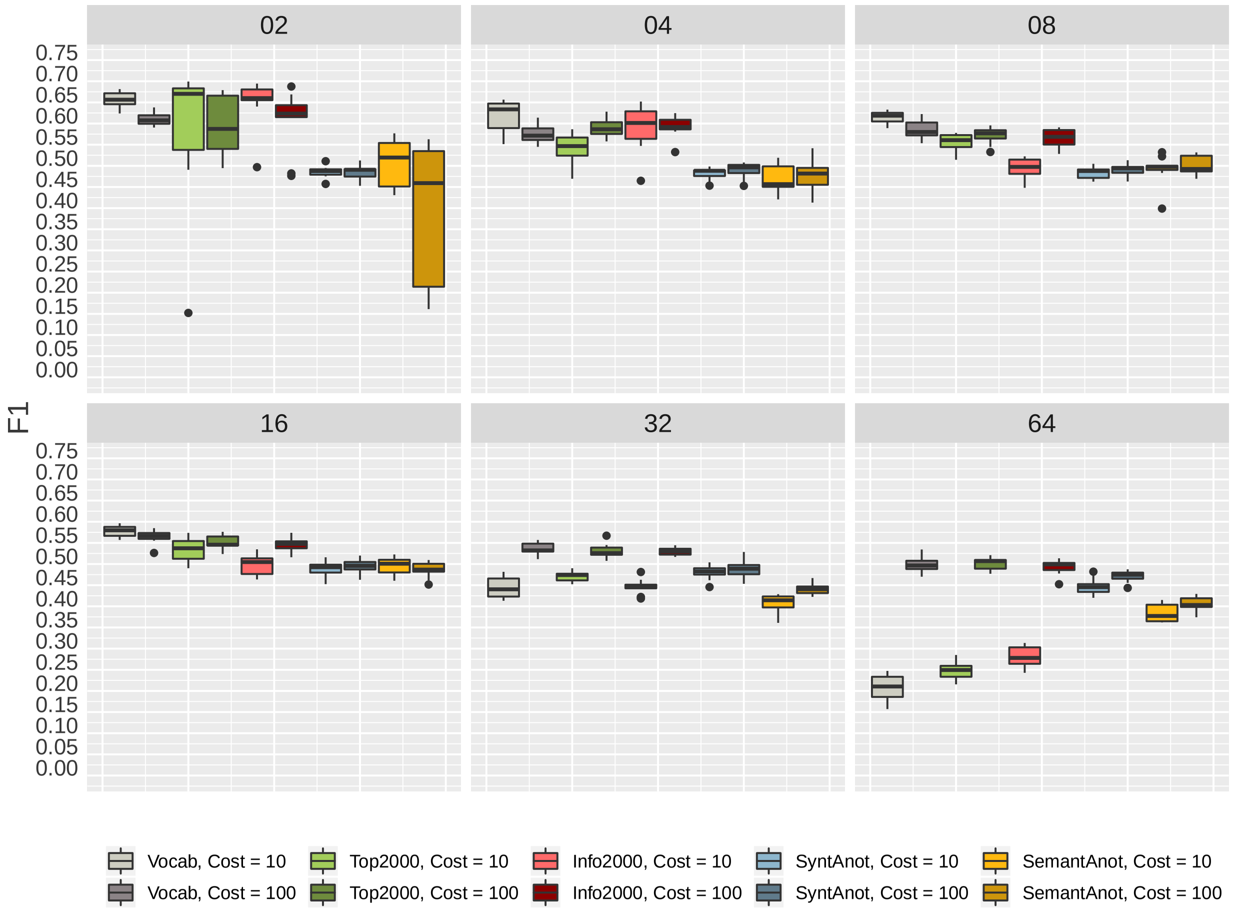


Figure 14: F1 observed for classifiers induced with SVM technique for costs 10 and 100 in a 10 fold cross validation scheme for scenarios with Vector Space Model text representation.

Figure 14 clarifies that, as we increase the number of groups we observe a correspondent decrease in classifier predictive power - measured by F1 - a fact that can be noted by the boxes heights decrease. This behaviour is expected once breaking the classifier induction computational effort implies a reduction in available information for the learning process.

We conclude, as well, that classifiers with cost 100 behaves better than cost 10, once for almost all text representation types dark tones boxes are higher than bright colours boxes, except for Vocabulary text representation. Though we might call attention to the fact that the difference increases as the total number of groups increases. Vocabulary text representation presents better results for cost 10 in 04 scenarios in 06. The scenarios in which this behaviour does not apply are the ones which mimic contexts with many computers available (32 and 64 boxes) not corresponding to scenarios which this methodology were thought to better suit (usual desktop computers).

Once Syntactic and Semantic annotation boxes define the lower boxes we see that using only external information to represent documents does not correspond to an effective way of inducing classifiers, in comparison with the other text representations analyzed.

Focusing attention in results for cost 100, we can see classifiers induced with the Top 2000 terms (top2000) have almost the same prediction power that classifiers induced with 2000 tokens chosen with Information Gain metric (info2000). Since defining top terms in a text collection is an easier task (with respect to computational effort) than defining the same amount of meaningful tokens using Information Gain we consider the former approach better than the latter.

As a consequence, whenever comparing results with Naive approach henceforth in this text, we will use results of Vocabulary with cost 10 or 100 and Top 2000 terms with cost 100.

The results related with the presented methodology are presented in Figures 15 and 16. The former enables the visualization for scenarios lines 12, 13 and 14 whilst the latter for lines 16, 17 and 18 in Table 5.7.4.1. Once cost 10 was clearly better than cost 100 we focus our attention in this results for taking 10 as cost parameter. Figures 22 and 23 in Appendix (Chapter 7) presents the results for both cost values for Syntactic and Semantic annotation, respectively.

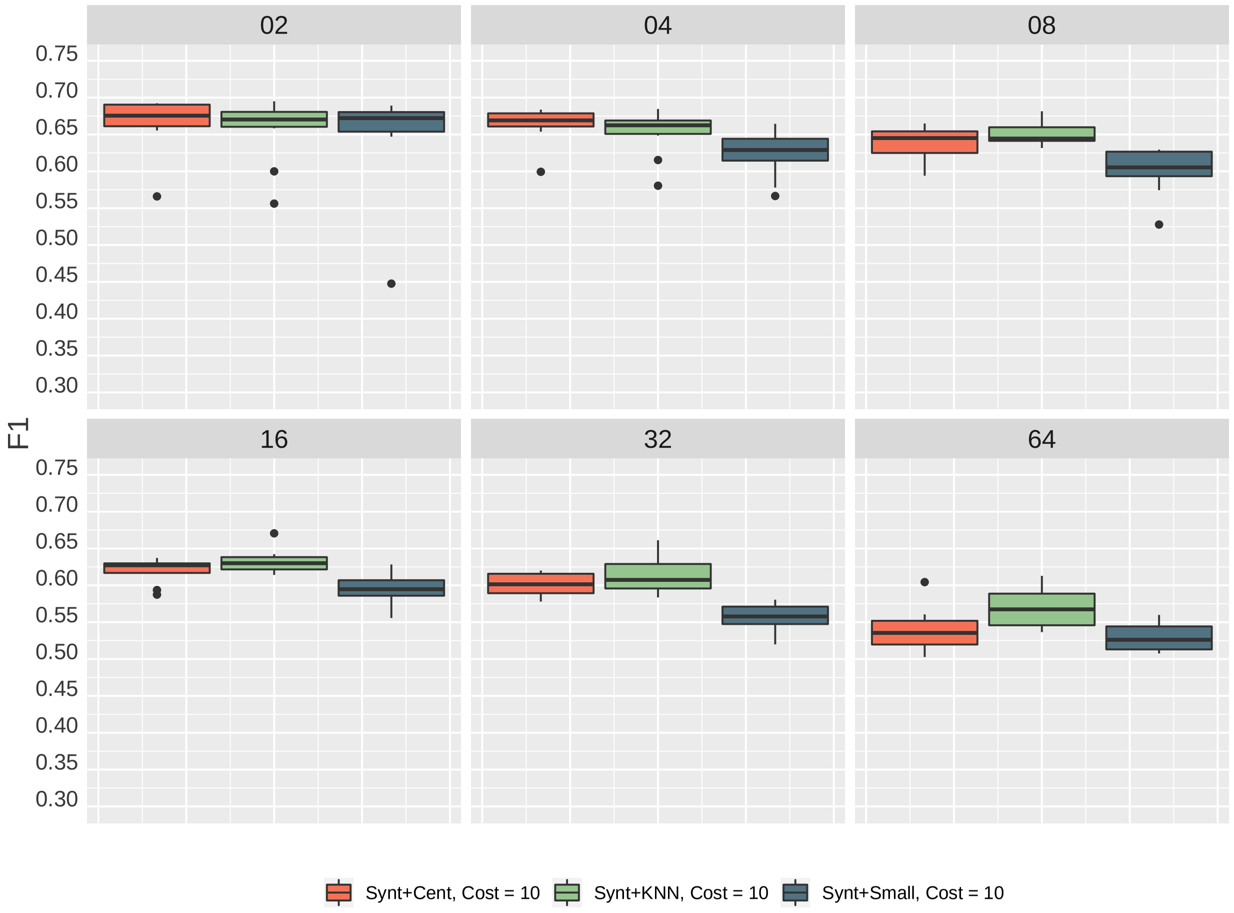


Figure 15: F1 observed for classifiers induced with SVM technique for costs 10 and 100 in a 10 fold cross validation scheme for scenarios with Syntactic Annotation in Sequence text representation.

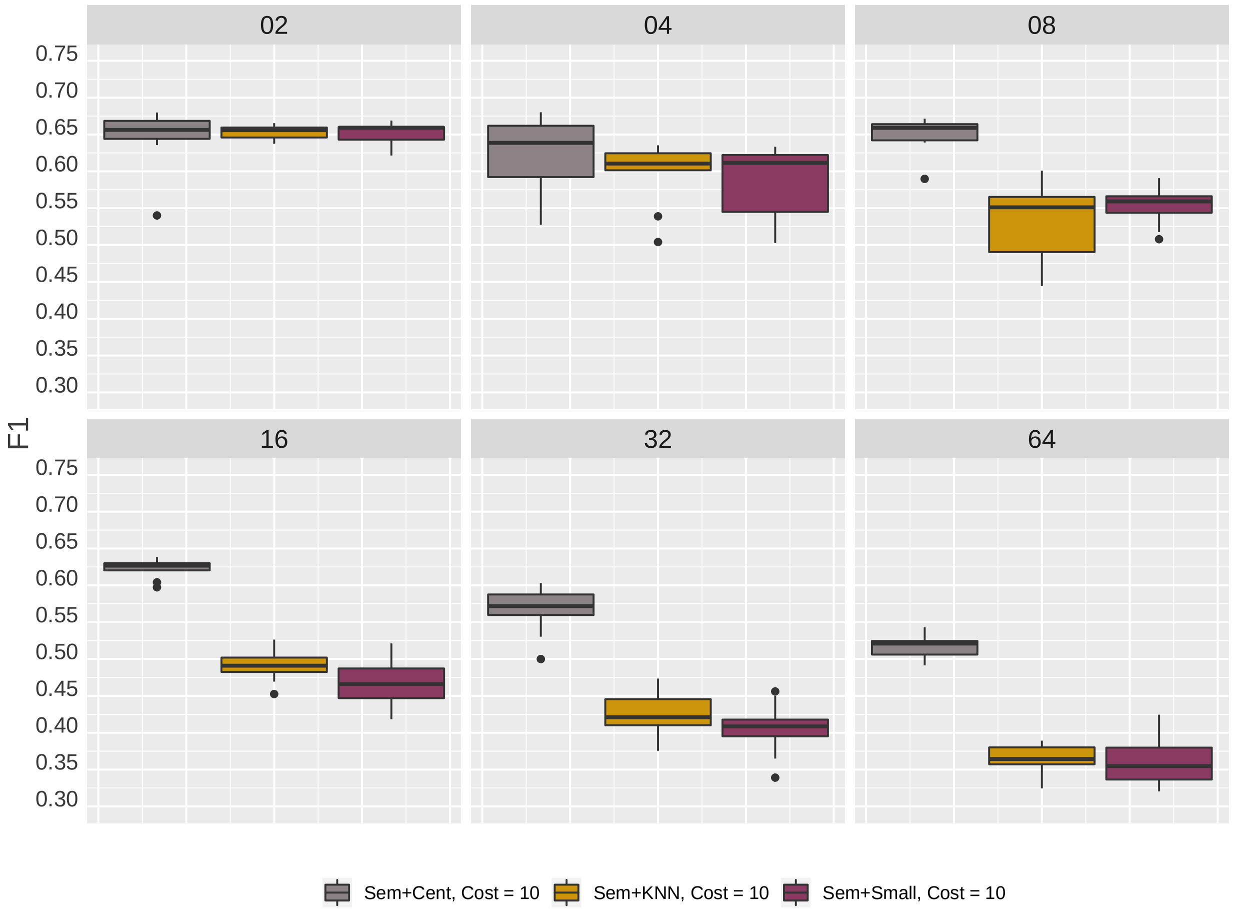


Figure 16: F1 observed for classifiers induced with SVM technique for costs 10 and 100 in a 10 fold cross validation scheme for scenarios with Semantic Annotation in Sequence text representation.

Figures 15 and 16 inspection allow us to immediately conclude that Smallest Mean Distance is the non-previously observed instance assignment which results in worst predictive power. However it is clear that the annotation nature plays an important role in this scenario. KNN assignment performs much better in Syntactic annotations scenarios than in Semantic ones.

Is is also interesting to observe that Centroid seems to be the instance assignment strategy with best results for Semantic annotation. Nevertheless, for Syntactic annotation, its superiority is clear only for small number of groups. As the total number of clusters increases KNN is the instance assignment scheme which corresponds to the best predictive power.

Another possible visualization corresponds to a plot which enables comparisons between Syntactic and Semantic annotation directly. Figure 17 serves that purpose. Aspects as the inverse relation between number of clusters predictive power are exalted once more. We also see that differences in F1 metric between text annotation types are lower for Centroid test instance assignment. The difference in behaviour for KNN and Smallest Mean Distance seem to be the same though KNN boxes are centered in bigger F1 values than Smallest Mean Distance.

Based on these results we can conclude that the proposed methodology performs better when using Syntactic text annotation as feature space. Given Syntactic text annotation, it is not possible to derive another straightforward conclusion with respect to non previously observed instances assignment. As both ways seem to have good results for predictive power, non previously observed instances assignment technique might be chosen by a second aspect: computational effort. By this we mean memory usage and/or total time needed to accomplish the task. When the user needs to prioritize predictive power KNN approach is recommended. When computational effort must be limited, Centroid approach is definitely the best one.

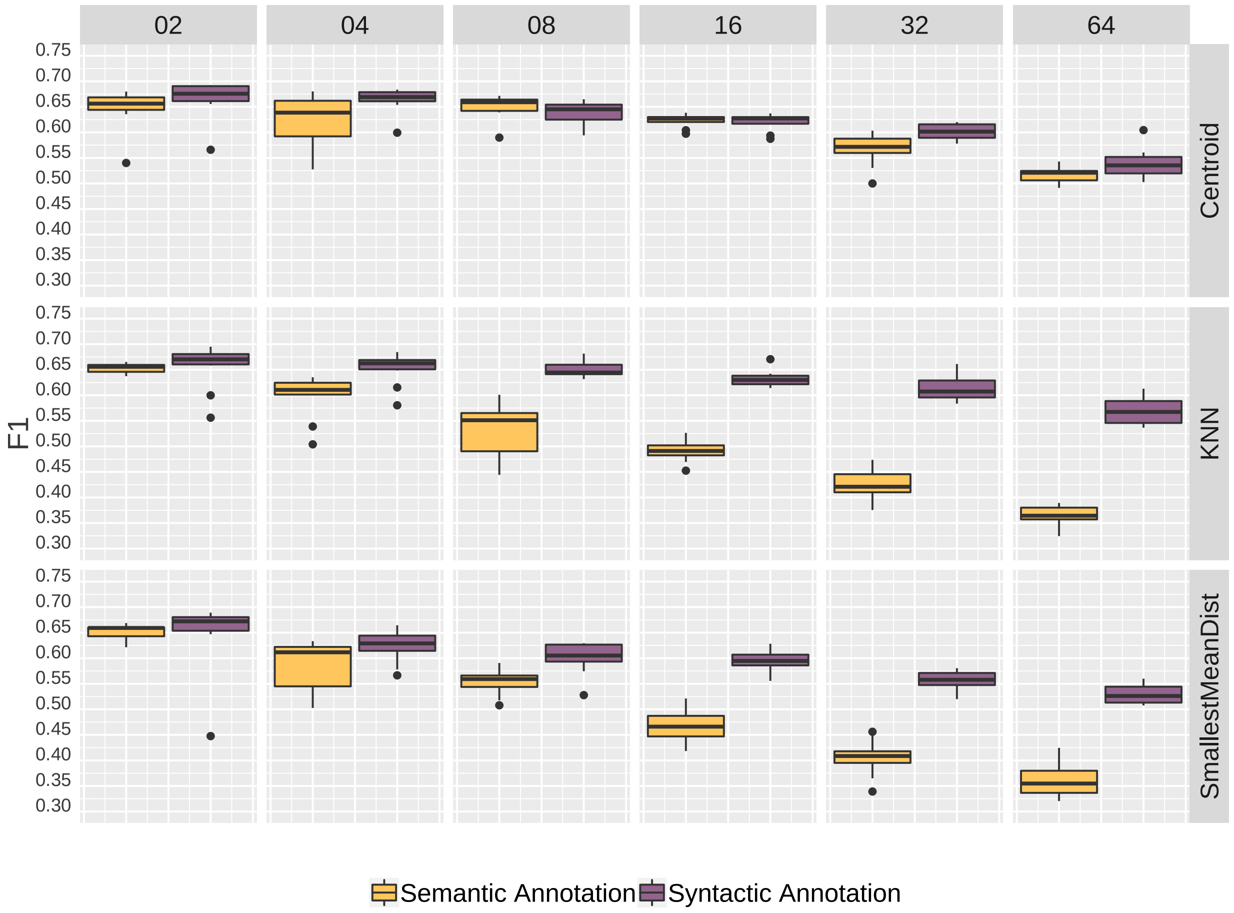


Figure 17: F1 observed for classifiers induced with SVM technique for cost 10 in a 10 fold cross validation scheme for scenarios with Syntactic and Semantic Annotation in Sequence text representation.

Since we know the best results related to the proposed methodology let's compare them with naive approaches. Figure 18 plots the best F1 results for Syntactic Annotation described previously (analyzing Figure 17) correspondent to scenarios 12 and 14 in Table 5.7.4.1. As a baseline scenario to compare results there are results for Vocabulary and 2000 Top terms text representation with Random Split Grouping method correspondent to scenarios 01 and 05 in Table 5.7.4.1 and derived from Figure 14 analysis.

Figure 18 clearly calls attention to the superiority of the proposed methodology results. Mustard and purple boxes correspondent to the usage of Syntactic Annotation for grouping instances are consistently higher that the others for all number of groups evaluated. As a consequence we can conclude the efficiency of the proposed methodology. We also see that 2000 top terms approach behaves a little worst than other naive approaches for 02, 04 and 08 groups. When analyzing 16, 32 and 64 groups the worst results correspond to Vocabulary test instance assignment with SVM cost parameter 10. This make explicit the need to define a cost parameter in each case, which is equivalent to the usage of a tuning parameter process.

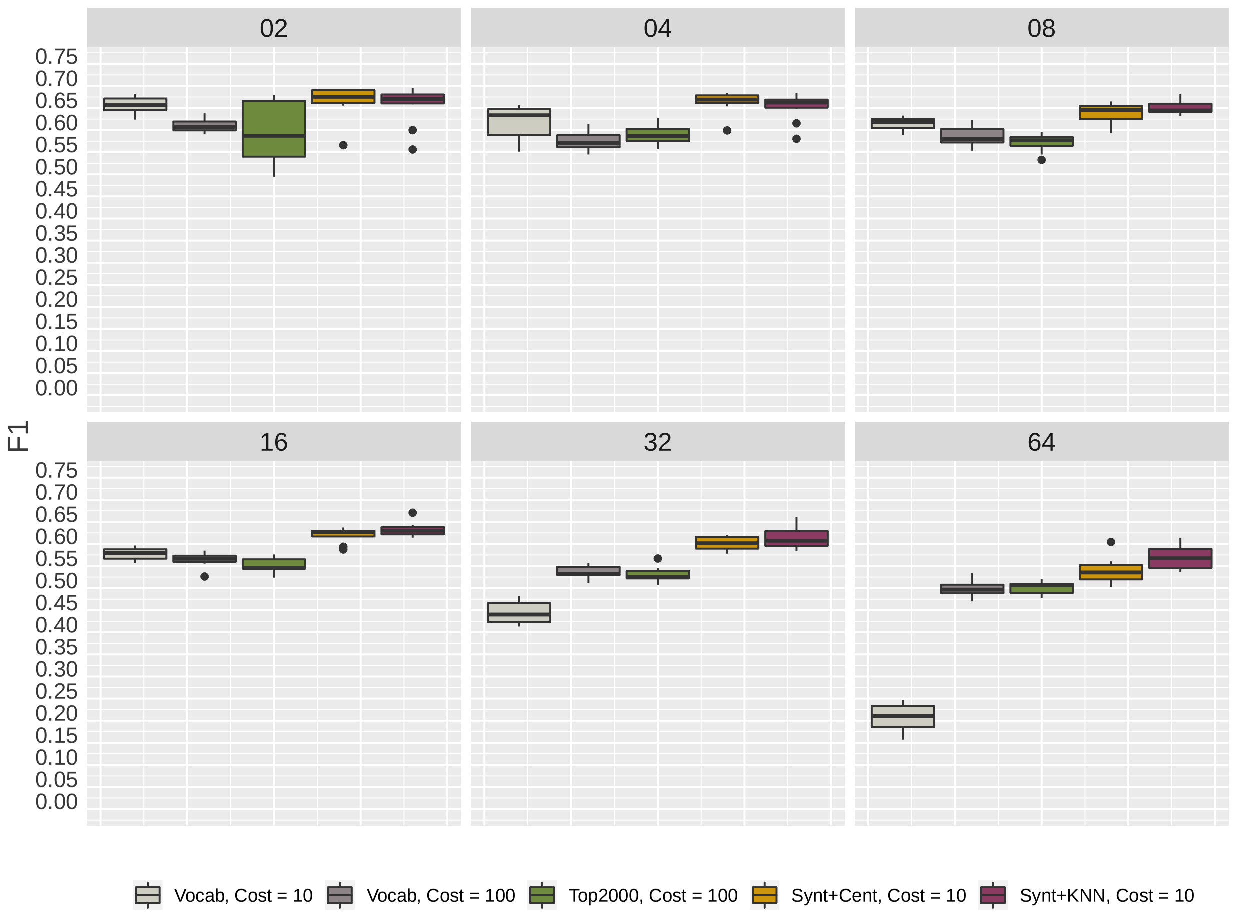


Figure 18: F1 observed in 10-fold cross validation for Best Classifiers.

5.9 Is it worthy to use the methodology?

We know that using the methodology implies some loss of information. Simplifying the feature space or the amount of instances used in the classification process clearly diminishes the algorithm learning power and as consequence it decreases the classifier's predictive power.

Hence, the loss associated with the methodology needs to be investigated and understood. With this in mind, we conducted an experiment with the following steps:

1. we obtain the F1 metric associated with the full data set and its whole vocabulary (unless a simple cleaning process). Once the classifier is inducted with full amount of information we say this is the maximum F1 value, $F1_{max}$.

2. we obtain the F1 metric for each number of groups of interest (02, 04, 08, 16, 32 and 64). We call these values as $F1_{02}, ..., F1_{64}$ observed.

3. we calculate the relation between the observed values and the maximum F1:

$$\frac{F1_{02}}{F1_{max}}, \frac{F1_{04}}{F1_{max}}, \frac{F1_{08}}{F1_{max}}, \frac{F1_{16}}{F1_{max}}, \frac{F1_{32}}{F1_{max}}, \frac{F1_{64}}{F1_{max}}. \tag{2}$$

For a better comprehension we observe these relations in a line plot, which is presented in Figure 19. Axis-x corresponds to the number of groups analyzed and axis-y represents the relation between F1 observed and $F1_{max}$.

We have another aspect which can hugely influence the results as well: the text representation. As a consequence, the experiment was thought varying also this aspect. We considered both Syntatic annotation and Semantic annotation taken as a sequence with KNN as a way to assign non-previously observed instances. We also included a scenario taking a *naive* text representation approach which is defined as a simple random split of the instances. Each text representation corresponds to a line/colour in Figure 19.

We also varied another important aspect: classification method adopted. Our aim with this is to check if the choice of the induction method is relevant or not to the conclusions obtained by the methodology. Hence, we tried four distinct methods: SVM, Logistic Regression, Random Forest and Boosting.

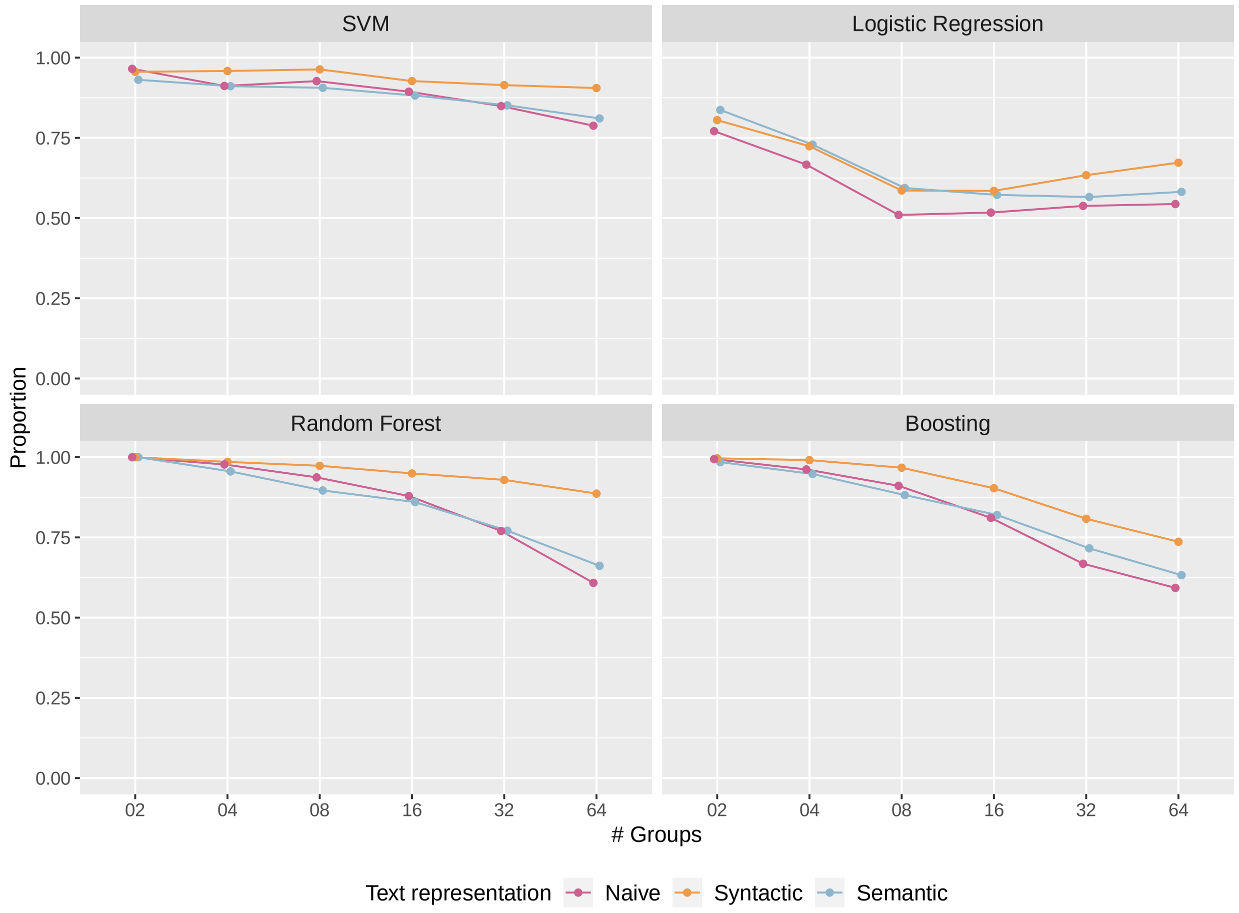


Figure 19: Relation between observed F1 and maximum F1 observed (full data set and whole vocabulary) by number of groups considering three distinct text representations and four different classification techniques.

Observing Figure 19 we can conclude that the proposed methodology has a better behaviour than the simple *naive* approach for all classification methods adopted. We can see that the pink line (*naive* way of clustering instances) is the bottom of the lines set. This means that, for all numbers of groups evaluated (values in axis-x) and all classification techniques studied (distinct lines), the loss in predictive power is lower when adopting the proposed methodology (yellow and blue lines).

In other words, we are aware that using the full data set and whole vocabulary observed corresponds to the complete source of information available. However, if it is needed to use less information (to reduce the feature space and to group the instances), it is possible to use our methodology, which decreases the loss of information. It also leads to a classifier with still good predictive power as we can see by the reduction of the ratio metric.

5.10 Distinct domain and language

This section is devoted to answer the question which is the methodology behaviour when exposed to other contexts/domains. After all, it is indispensable that the methodology does not vary as a function of the topics it is being applied.

We chose to work with a well-known data set, developed by Go et al. [2009] called Sentiment140. It consist of a collection of documents collected on Twitter with respect to an usual sentiment scale: negative, neutral and positive. Sentiment140 data set was designed using a distant supervision approach that replaces usual manual labelling process by a time saving one. It was conceived collecting tweets with emoticons listed in a pre-specified negative and positive emoticon sets.

Sentiment140 suits the question being investigated since it corresponds to a big data set, with initially 1,600,000 (a million and six hundred thousand) distinct documents. Since it also comes from Twitter, it is composed by small documents (constrained in size). Differently from previous analysis, these documents were written in English. The data set used in this analysis is available at Kaggle web page [kaggle, 2019].

5.10.1 Kaggle data set characteristics

An initial inspection in the Kaggle version data set revealed the existence of only positive and negative polarities. Also, we observed 1,685 instances with two different labels associated and henceforth these instances were removed from the analysis.

Tweets were pre-processed as described in Go et al. [2009]. Texts were converted to lower case, users references were replaced by a single pattern, links were replaced by a single pattern, more than two equal consecutive letters were replaced by two equal consecutive letters, punctuation marks were removed, words of a single character were removed, shift and tabs were replaced by single space.

An example of negative polarity tweet is the sentence *I hate when I have to call and wake people up*, posted in *Mon Apr 06 22:20:44 PDT 2009* by *ChicagoCubbie* (ID: 1467814438).

A positive polarity tweet instance is *im meeting up with one of my besties tonight! Cant wait!! - GIRL TALK!!*, posted in *Mon Apr 06 22:22:45 PDT 2009* by *becca210* (ID: 1467822273).

5.10.2 Sequence of *part-of-speech* annotation

As part of the methodology we annotated each instance terms with its respective *part of speech* tags using Linguakit tool [Cillenis, 2017a]. Instances which the tool was not able to work properly were simply neglected due to the enough amount of information available and the low benefit of investigating small amount of problematic instances characteristics. It resulted in 1,554,249 documents or, in other words, we came up with an initial set of 1,554,24 *part of speech* annotated documents represented as sequences to cluster. At this point, the data set was made of 774,366 negative polarity instances and 779,884 positive polarity instances. In relative frequency, it is 0.4982 and 0.5017 of negative and positive instances, respectively, which denotes a balanced labeled scenario, in distinction from the crime scenario presented before.

5.10.3 Computer characteristics

This last experiment was conducted in a computer with Ubuntu 18.04.2 LTS System, 15,5GiB of memory, Intel Core i7 CPU 975 @3.33 GHz x 8 processor, 64 bit and 491,2 GB disk, 08 cores - a regular Desktop computer, nowadays. Thus, the experiment is conducted in a scenario of a simple user, someone who does not have access to a super computer with advanced settings.

5.10.4 CLARA cluster adaptation implementation

At this point, the main aim is to group similar documents based on Levenshtein distances between sequences of *part-of-speech* annotated tags.

We divided the data set in two main parts: one for training phase and one to assess final classification quality, test set. The former is a random sample of size one million whilst the latter is the remaining.

Clustering a million instances data set imposes innumerable challenges. Even considering recent clustering implementations it corresponds to a huge computational effort. As a consequence, we derived our own clustering implementation, being another contribution of this work.

It is inspired by CLARA (Clustering Large Application) clustering technique, presented in Aggarwal and Reddy [2013]. However, instead of choosing the setting with lower overall dissimilarity, we choose the most homogeneous setting with respect to the amount of instances in each group since we desire data sets homogeneous in size to decrease the task effort and, as a consequence, its computational time (sub-section 5.1).

Just as CLARA algorithm, we sample a certain amount of instances at random. With this smaller subset D, we obtain Levenshtein distance matrix with *stringdist* library [van der Loo, 2014]. Following, we cluster instances in k groups using PAM (Partition Around Medoids) technique from *cluster* library [Maechler et al., 2019]. Each of the remaining data set is assigned to one of the defined medoids whose Levenshtein distance is the minimum observed. These steps are replicated m times. We choose the medoid setting which returns the most balanced instances assignment. For this analysis, we took samples of size $D = \mathit{20{,}000}$ (remaining data set of 980,000) and used $k = \mathit{7}$ groups, $m = \mathit{50}$ times. We used *data.table* library [Dowle and Srinivasan, 2019], well known to store and manipulate huge data sets.

We clarify that the contribution does not reside solely on choosing the most balanced instances assignment. Once the Levenshtein metrics and cluster definitions are computationally costly, we used parallel implementations for distance metrics and cluster definitions and we did it on BATCH mode, each replica at a time turning R on and off automatically each replica.

Figure 20 exemplifies six out of the total fifty clustering results. We see bar plots representing the relative frequencies by group. Frames are labeled with the iteration number. Upper frames bring the two most balanced groups, which we can confirm noticing that the bigger bars are around 0,15. Frames in the middle describe scenarios with more non-homogeneous instances assignment. Lower frames show two unbalanced configurations with one group reaching almost 20% of the instances. We chose replica 03 since it defines the most balanced clusters. The clustering phase took approximately 2.708889 hours to run.

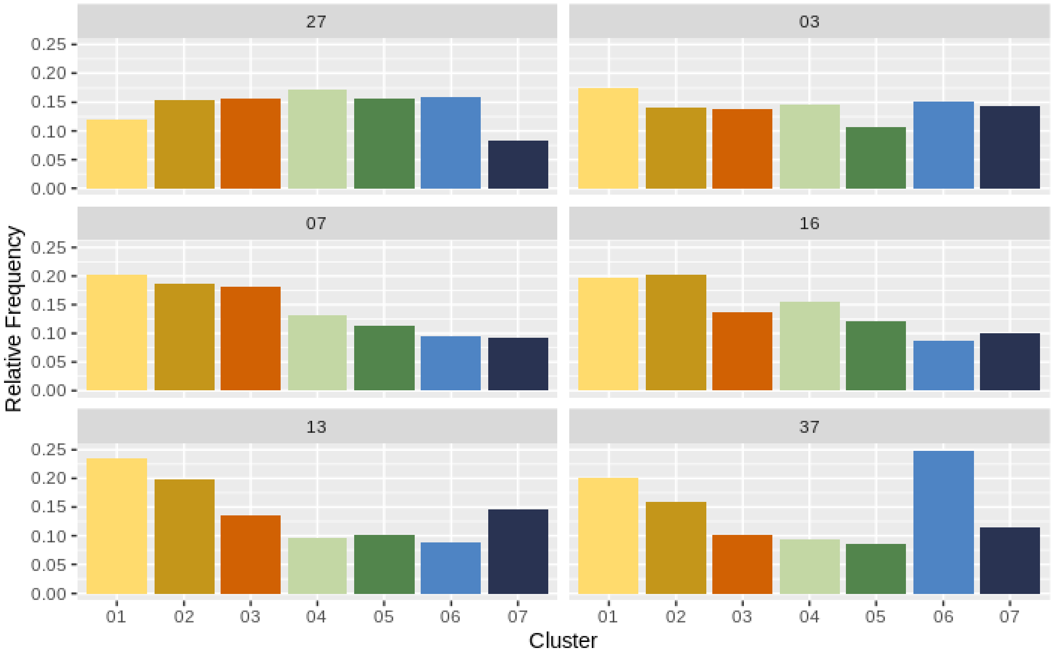


Figure 20: Percentage of instances assigned to 07 groups in adapted CLARA clustering for six out of fifty replicas: Upper frames bring the two most balanced groups. Middle frames describe scenarios with more non-homogeneous instances assignment. Lower frames show two unbalanced configurations.

5.10.5 Feature Selection for Classification phase

As mentioned previously, it is also a big challenge to represent instances with meaningful features in the context of text data. This step is dedicated to save time and effort in classifier induction process. The initial sample of 1,000,000 instances was represented by a collection of 81,375 features/terms after applying pre-processing steps as described in sub-section 5.10.1.

Based on a simple analysis, we decided to remove *stop words*. Once the data set is in English we used a *stop words* collection of *stopwords* library [Benoit et al., 2017], which comprises 571 terms leading to a data set with 81,004 terms. An inspection of term frequency distribution showed its huge asymmetry and the presence of many terms with really low frequency. As a consequence we decided to maintain only features with at least 50 occurrences, resulting in 8,794 distinct instances.

Information Gain feature selection technique was applied at this stage. We used *FSelectorRcpp* library implementation [Zawadzki and Kosinski, 2019]. Although the referred library states that it has an optimized implementation, it requires data set to be in columnar (or wide) format which demands bigger memory availability. We also worked around this issue and this is considered a contribution. Once Information Gain is part of univariate feature selection methods we did it splitting the data set by terms. To correctly compute each feature metric we imputed data so that, we count the number of its occurrences in all documents. This process took 1,85 hours to run. The algorithm returned 1,751 features/terms with importance different of zero, so that the algorithm reveals 20% out of the 8,794 are useful describing the target/label. Observing some features/terms with numbers/digits we excluded other 31 features and the final data set is represented by 1,720 terms. The seven Document-term matrices to induce classifiers with the selected features were obtained in 1.55 hours.

5.10.6 Classification results

Classifier induction was done with SVM technique exploring three distinct cost values: 0,1, 1 and 10 with *LiblineaR* library [Helleputte, 2017].

We assessed the results with F1 harmonic mean of precision and recall, considering the positive polarity as the reference class. The F1 distribution results, by cost value, are described in Figure 21 which clearly enables to conclude that the classifier has no distinct behaviour as a function of the cost parameter. The overall precision, summing up Confusion Matrices entries for all clusters/classifiers, is around 0.76 and recall is around 0.81 which results in F1 around 0.78.

Although the classifiers are the same with respect to their quality their induction process time differs considerably. For cost 0,1, the whole process took 59.66 minutes. Nevertheless, cost 1 experiments took 7.63 minutes almost the same as cost 10 experiments which took 7.53 minutes.

Thus, we can conclude that inducing classifiers with cost 10 we obtain satisfactory classifiers with respect to their quality in less than 08 minutes. In other words, the proposed methodology was proven to work in a distinct domain (sentiment analysis in opposition to crime from the initial experiments) and language (English differently from Portuguese from the beginning analysis). This experiment is a challenge since we used a huge collection of text, around 1,000,000 instances to train a classifier and 500,000 instances to be predicted. We reinforce its success once it would not be able to directly perform such analysis using an usual methodology and its available implementations in simple Desktop computers. Therefore, our proposal empowers simple users to induce classifiers for large text data sets in a reasonable time without loosing the prediction power.

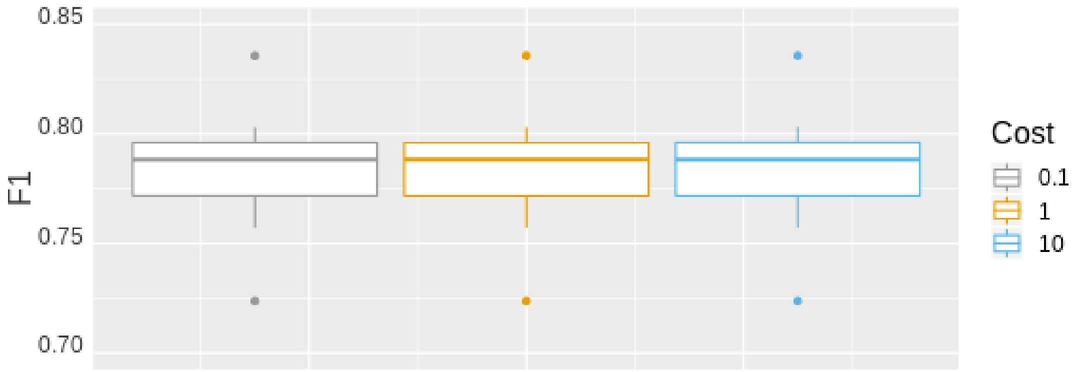


Figure 21: F1 metric for lineat SVM classifiers induced for cluster replica 03.

6 Final remarks

In this work we proposed a new methodology to enable to induce classifiers for large data sets of small texts (constrained in size). Our approach is devoted to simple users, whose computers are usual desktops and without advanced computational abilities, as distributed or gpu programming.

The methodology is divided in two steps. In the first moment, the whole data set is splitted into separated smaller data sets. Following, for each smaller data set a classifier is induced. Hence, the large initial effort is replaced by smaller efforts, something usually known by divide-and-conquer strategy.

The utility and usability of the methodology directly depends on the balance of the smaller (splitted) data sets with respect to its number of instances. The overall task effort actually decreases if, and only if, the splitted data sets define classifier induction tasks of approximately same effort. We conceived same size groups using a simple clustering technique (PAM) and an available similarity measure for texts (Levenshtein). Using external text information, Syntax part-of-speech tags Annotation represented in sequences we define balanced groups in size which, as a consequence, enable to induce classifiers with approximately equal effort. Representing texts by syntax part-of-speech tags annotation in sequences is a contribution of this work.

The classifier induction process is done with the observed vocabulary as features, taking just its useful set which is defined with feature selection techniques. By doing this, we reduce computational effort also saving processing time.

All methodology steps were thought to be performed with parallel computing. All experiments presented were accomplished with parallel computing in usual multi core Desktop computer.

We used a popular data analysis software (R) and available implementations (for KMedois, Levenshtein distance metric for strings, classification techniques and analysis) of almost all methods implied, except for some little modifications done to overcome technical limitations or to speed up the tasks. It empowers simple users, without advanced coding skills to analyze this kind of data.

We also add the fact that the methodology was performed with a free text annotation tool (Linguakit) showing not to require expensive/costly infra structure.

The methodology is applicable to different classifier induction techniques (SVM, Random Forest, Bagging, Boosting, Logistic Regression, among others), giving flexibility to the user in the pursuit of good results and proving to suit different data types.

The methodology was proven to work for different domains, criminology and sentiment analysis. In other words, it is applicable to distinct contexts.

The presented method also worked for distinct languages, (Portuguese and English were explored in this text) as long as there is Syntax annotation tool available.

As a future work we aim to minimize an undesirable and common aspect in text analysis. As previously mentioned, fads, slang and posts are strongly related to time moment. It is a really common aspect because informality, gossip and the massive sharing of a single content (turning it the so called web virus) are really strong characteristics in the web context. In de Oliveira et al. there is a good example of how a virus information can change a tweet based analysis. We say that problems whose properties change over time have a concept drift, something strongly studied nowadays like Nishida et al. [2012] work.

Our aim is to continuously feed the Train data base making it robust to fads, slang and posts specific from a given time moment. We want to add to the Training data base posts already classified by the proposed method which have high probability of being correct. This would continuously improve the technique without the external expert agent requisition. For this, it would be necessary to develop a metric to measure the quality or trust in the label given by the classifier. We should add the need to specify how many new documents would enter the Training base and if old documents should get out of it. Besides it, we must define the temporal mechanism adopted and all these aspects increment the proposed methodology.

7 Appendix

Agredir	Clonaram	Homicidios	Roubadas
Agrediram	Clonou	Ladra	Saida bancaria
Agrediu	Estelionatario	Ladras *	Saida de agencia
Agressao	Estelionatarios	Ladrao	Saida de banco *
Agressor	Estelionato	Ladroes	Saidinha bancaria
Agressores *	Estuprador	Latrocinio	Saidinha de agencia
Arrastao	Estupradores	Mata	Sequestra
Arrombamento	Estupro	Mataram	Sequestrador
Assalta	Estupros	Matou	Sequestradores *
Assaltada	Estuprada	Morreu	Sequestraram
Assaltadas	Estupradas	Morreram	Sequestro
Assaltado	Estupraram	Morto	Sequestros
Assaltados	Estuprou	Mortos	Sequestrou
Assaltante	Fraudaram *	Morta	Sequestrado
Assaltantes	Fraude	Mortas	Sequestrados
Assaltaram	Fraudes	Pedofilia	Sequestrada
Assalto	Fraudou *	Pedofilo	Sequestradas
Assaltos	Furta	Pedofilos	Tarado
Assaltou	Furtaram	Pornografia	Tentativa de assalto
Assassina	Furto	Presa	Tentativa de furto *
Assassinas	Furtos	Preso	Tentativa de homicidio *
Assassinada	Furtou	Quadrilha	Tentativa de roubo *
Assassinadas	Furtada	Quadrilhas	Tentativa de sequestro *
Assassinado	Furtadas	Refem	Traficante
Assassinados	Furtado	Refens	Traficantes
Assassinaram	Furtados	Rouba	Traficaram *
Assassinato	Gangue	Roubaram	Trafico
Assassinatos	Gangues	Roubo	Traficos
Assassino	Golpe	Roubos	Traficou *
Assassinos	Golpes	Roubou	Violencia domestica
Assassinou	Golpista	Roubado	Vitima
Atentado violento ao pudor	Golpistas	Roubados	Vitimas
Clonagem	Homicidio	Roubada	

Table 7.0.0.1: Terms used as keywords to collect data. The colors represent general type of crime, in Portuguese. Blue: *contra a pessoa*, Green: *contra o patrimônio*, Red: *sexuais* and Purple: general expressions. Terms marked with ∗ are not common in small cities.

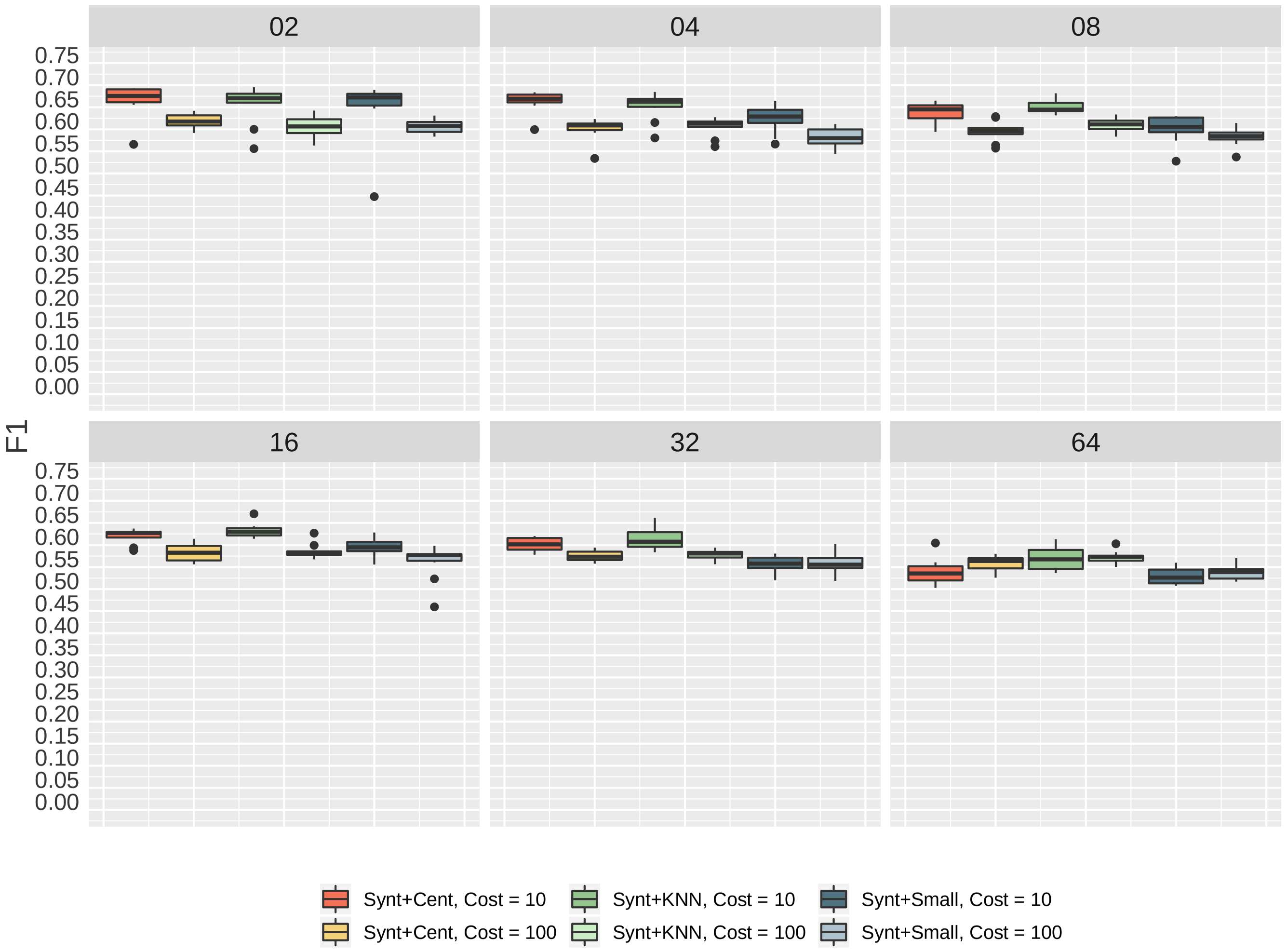


Figure 22: F1 observed for classifiers induced with SVM technique for costs 10 and 100 in a 10 fold cross validation scheme for scenarios with Syntactic Annotation in Sequence text representation.

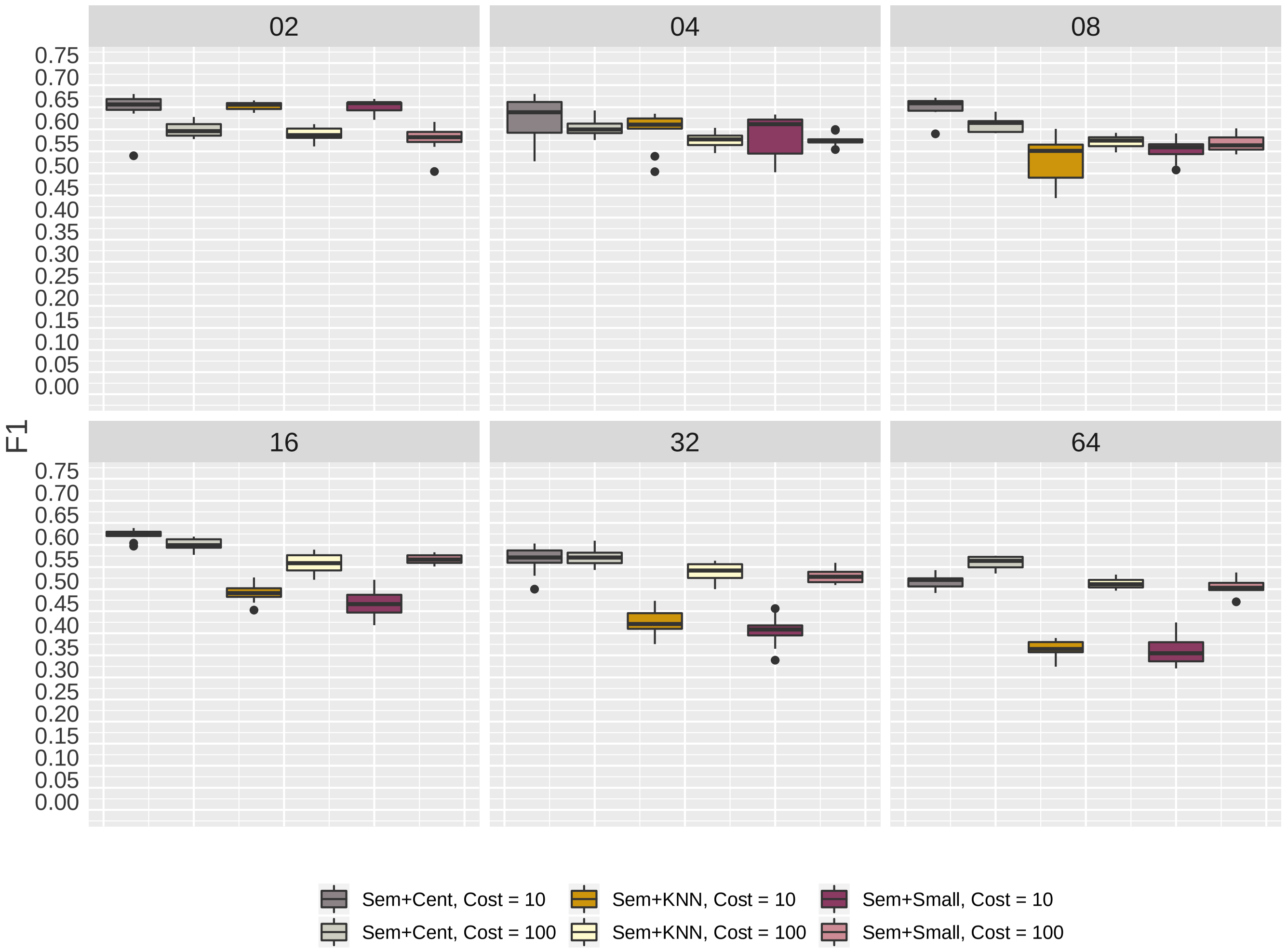


Figure 23: F1 observed for classifiers induced with SVM technique for costs 10 and 100 in a 10 fold cross validation scheme for scenarios with Semantic Annotation in Sequence text representation.

References

C. C. Aggarwal and C. K. Reddy, editors. *Data clustering: algorithms and applications*. CRC press, 2013.

T. J. Alelyani, Salem and L. Huan. *Feature Selection for Clustering: A Review*, pages 30–55. Volume 1 of Aggarwal and Reddy [2013], 2013.

R. Analytics and S. Weston. *iterators: Provides Iterator Construct for R*, 2018. URL https://CRAN.R-project.org/package=iterators. R package version 1.0.10.

J. B. Arnold. *ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'*, 2018. URL https://CRAN.R-project.org/package=ggthemes. R package version 4.0.1.

K. Benoit, D. Muhr, and K. Watanabe. *stopwords: Multilingual Stopword Lists*, 2017. URL https://CRAN.R-project.org/package=stopwords. R package version 0.9.0.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

C. Bouras and V. Tsogkas. A clustering technique for news articles using wordnet. *Knowledge-Based Systems*, 36:115–128, 2012.

Cillenis. https://linguakit.com/pt/. online, 2017a. Acessed in July/2017.

Cillenis. https://talp-upc.gitbooks.io/freeling-user-manual/content/tagsets/tagset-pt.html. online, 2017b. Acessed in July/2017.

Cillenis. https://http://cilenis.com. online, 2018a. Acessed in July/2018.

Cillenis. https://talp-upc.gitbooks.io/freeling-4-0-user-manual/content/tagsets/tagset-pt.html. online, 2018b. Acessed in June/2018.

M. Corporation and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2017. URL `https://CRAN.R-project.org/package=doParallel`. R package version 1.0.11.

A. T. David C Anastasiu and G. Karypis. Document clustering: the next frontier. In C. C. Aggarwal and C. K. Reddy, editors, *Data clustering: algorithms and applications*, chapter 13, pages 305–328. CRC press, Oxford, 2013.

D. M. de Oliveira, R. C. Souza, D. E. de Brito, W. Meira Jr, and G. L. Pappa. Uma estratégia não supervisionada para previsão de eventos usando redes sociais.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. Misc functions of the department of statistics (e1071), tu wien. *R package*, pages 1–5, 2008.

L. S. F. C. dos Santos. Estudo online da dinâmica espaço-temporal de crimes através de dados da rede social twitter. Master's thesis, Departamento de Estatística - Universidade Federal de Minas Gerais, Belo Horizonte, MG - Brasil., 2015.

M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2019. URL `https://CRAN.R-project.org/package=data.table`. R package version 1.12.2.

N. Du, M. Farajtabar, A. Ahmed, A. J. Smola, and L. Song. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 219–228. ACM, 2015.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9 (Aug):1871–1874, 2008.

I. Feinerer and K. Hornik. *tm: Text Mining Package*, 2018. URL `https://CRAN.R-project.org/package=tm`. R package version 0.7-5.

A. Gabadinho, G. Ritschard, N. S. Mueller, and M. Studer. Analyzing and visualizing state sequences in r with traminer. *Journal of Statistical Software*, 40(4): 1–37, 2011.

P. Gamallo and M. Garcia. Linguakit: uma ferramenta multilingue para a análise linguística e a extração de informação. *Linguamática*, 9(1):19–28, 2017.

M. Garcia and P. Gamallo. Yet another suite of multilingual nlp tools. In *International Symposium on Languages, Applications and Technologies*, pages 65–75. Springer, 2015.

A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

T. Helleputte. *LiblineaR: Linear Predictive Models Based on the LIBLINEAR C/C++ Library*, 2017. R package version 2.10-8.

J. Huling. *jcolors: Colors Palettes for R and 'ggplot2', Additional Themes for 'ggplot2'*, 2018. URL `https://CRAN.R-project.org/package=jcolors`. R package version 0.0.3.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

D. Jurafsky. *Speech & language processing.* Pearson Education India, 2000.

kaggle. https://www.kaggle.com/kazanova/sentiment140. online, 2019. Acessed in June/2019.

A. Liaw and M. Wiener. Classification and regression by randomforest. r news. 2002; 2 (3): 18–22, 2016.

S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2019. R package version 2.0.9 — For new features, see the 'Changelog' file (in the package source).

Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct for R*, 2017. URL `https://CRAN.R-project.org/package=foreach`. R package version 1.4.4.

K. Nishida, T. Hoshide, and K. Fujimura. Improving tweet stream classification by detecting changes in word probability. In *Proceedings of the 35th international*

ACM SIGIR conference on Research and development in information retrieval, pages 971–980. ACM, 2012.

A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.

S. Piao, F. Bianchi, C. Dayrell, A. D'egidio, and P. Rayson. Development of the multilingual semantic annotation system.

S. S. Piao, P. E. Rayson, D. Archer, F. Bianchi, C. Dayrell, M. El-Haj, R.-M. Jiménez, D. Knight, M. Křen, L. Lofberg, et al. Lexical coverage evaluation of large-scale multilingual semantic lexicons for twelve languages. *Linguamática*, 2016.

J. Pustejovsky and A. Stubbs. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications.* " O'Reilly Media, Inc.", 2012.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL `https://www.R-project.org/`.

C. K. Reddy and V. Bhanukiran. *Feature Selection for Clustering: A Review*, pages 88–107. Volume 1 of Aggarwal and Reddy [2013], 2013.

M. A. H. Reddy, Chandan K. and Z. Mohammed. *Feature Selection for Clustering: A Review*, pages 382–405. Volume 1 of Aggarwal and Reddy [2013], 2013.

G. Ridgeway et al. gbm: Generalized boosted regression models. *R package version*, 1(3):55, 2006.

T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008.

J. Silge and D. Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016. doi: 10.21105/joss.00037. URL `http://dx.doi.org/10.21105/joss.00037`.

A. M. R. O. M. D. Souza, C. S. N. P. Roberto and W. Meira Jr. Infection hot spot mining from social media trajectories. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECMLP-KDD'16)*, pages 645–654. IEEE, 2016.

E. Stamatatos. Authorship attribution based on feature set subspacing ensembles. *International Journal on Artificial Intelligence Tools*, 15(05):823–838, 2006.

E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

D. Temple Lang and J. Wallace. *RJSONIO: Serialize R Objects to JSON, JavaScript Object Notation*, 2018. URL `https://CRAN.R-project.org/package=RJSONIO`. R package version 1.3-1.1.

A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185, 2010.

UCREL. http://ucrel.lancs.ac.uk/whatis.html. online, 2018a. Acessed in January/2018.

UCREL. http://ucrel.lancs.ac.uk/usas/. online, 2018b. Acessed in January/2018.

UCREL. http://ucrel.lancs.ac.uk/usas/gui/. online, 2018c. Acessed in January/2018.

UCREL. *USAS guide*. UCREL, Lancaster, UK, 2018d. URL `http://ucrel.lancs.ac.uk/usas/usas_guide.pdf`.

M. van der Loo. The stringdist package for approximate string matching. *The R Journal*, 6:111–122, 2014. URL `https://CRAN.R-project.org/package=stringdist`.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `http://ggplot2.org`.

H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2018. URL `https://CRAN.R-project.org/package=stringr`. R package version 1.3.1.

H. Wickham and L. Henry. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*, 2018. URL `https://CRAN.R-project.org/package=tidyr`. R package version 0.8.1.

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL `https://CRAN.R-project.org/package=dplyr`. R package version 0.7.6.

J. Yin and J. Wang. A text clustering algorithm using an online clustering scheme for initialization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1995–2004. ACM, 2016.

Z. Zawadzki and M. Kosinski. *FSelectorRcpp: 'Rcpp' Implementation of 'FSelector' Entropy-Based Feature Selection Algorithms with a Sparse Matrix Support*, 2019. URL `https://CRAN.R-project.org/package=FSelectorRcpp`. R package version 0.3.1.

A. Zimek. *Feature Selection for Clustering: A Review*, pages 201–220. Volume 1 of Aggarwal and Reddy [2013], 2013.