TÉCNICO LISBOA

PROLEGIS

Intelligent Search in Legislation Databases

Hugo Miguel de Jesus Lopes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Luís Manuel Marques Custódio
Prof. Carlos Alberto Pinto Ferreira

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Luís Manuel Marques Custódio
Member of the Committee: Prof. Ana Luísa Nobre Fred

May 2016

To my family and friends, specially the ones who had to endure me in my worst times.

Resumo

A legislação portuguesa, semelhante a outros países, não é publicada de forma organizada em tópicos e conceitos, mas sim organizada por um sistema numerado por ordem e data da publicação. Para um cidadão comum ou mesmo investigadores, procurar informação relativamente a um tema ou problema específico é uma tarefa árdua e complexa.

A categorização manual de texto legal, além de necessitar de profissionais especializados, é uma tarefa que requer bastante tempo devido ao elevado número de documentos existentes. Como tal, o objetivo deste trabalho centra-se em avaliar a possibilidade de automaticamente atribuir-se a estes documentos legislativos uma categoria. Dado a elevada quantidade de documentos, a tarefa da Classificação de Textos é normalmente apoiada em algoritmos de *Machine Learning*. Existem principalmente duas abordagens: a categorização supervisionada e não supervisionada. A classificação supervisionada atribui categorias pré definidas aos documentos, baseando-se em parâmetros estimados a partir de exemplos previamente classificados. A classificação não supervisionada utiliza as características intrínsecas dos documentos, sem necessitar de categorias previamente estipuladas ou documentos classificados.

O foco deste trabalho será no domínio supervisionado, ainda assim, uma análise de agrupamento não supervisionado será realizada. Múltiplos algoritmos de classificação são também experimentados, utilizando documentos pré-classificados, para avaliar comparativamente os seus desempenhos na classificação. *Support Vector Machine* (SVM), *K-Nearest Neighbours*, *Multinomial Naive Bayes* e *Decision-Trees* são analisados individualmente e, para procurar melhorar os resultados, em conjunto com várias técnicas de pré processamento de características. *Latent Semantic Indexing* (LSI), seleção de características com diferentes métricas e stemização são analisados.

Agrupamento com *K-means* não demonstrou a existência de uma estrutura subjacente que permitisse classificação de texto. O SVM demonstrou ser o melhor classificador, atingindo desempenhos semelhantes aos encontrados na literatura. A seleção de características com a métrica χ^2 permitiu atingir classificações próximas das experiências com todo o conjunto de características, utilizando apenas 20% destas.

O LSI demonstrou ser uma técnica de redução de dimensionalidade poderosa. Com uma redução de 95% das características, este método permitiu atingir, ainda que marginalmente, melhores resultados que com todo o conjunto de características. Stemização, pelo contrário, demonstrou apenas ser benéfica em *corpus* onde o número de características é reduzido. Estes métodos de pré processamento, ainda que não demonstraram melhorias como esperado, permitiram reduções significativas na dimensionalidade, e consequentemente, reduzir tempos de processamento mantendo resultados semelhantes.

Palavras-chave: Legislação Portuguesa, Textos Legais, Categorização de Texto, Classificação Supervisionada

Abstract

Portuguese legislation, similarly to other countries, is not published in an organized way, being it by topics or concepts. Instead, it is organized by a numbering system which follows the publication order. For a common citizen or even researchers, searching for information about a subject or a specific problem is an hard and complex task.

The categorization of legal texts, besides requiring specialized labour, is a task which would need a great amount of time due to the quantity of published documents. The purpose of this work focuses in evaluating the possibility of automatically assign to this legislative documents a category. Given the large amount documents, the Text Categorization task often relies on Machine Learning algorithms. There are mainly two approaches: the supervised and the unsupervised categorizations. The supervised categorization assigns predefined category labels to documents, based on the estimated parameters from a set of classified examples. The unsupervised categorization uses the intrinsic characteristics of the documents, without requiring predefined categories or classified documents.

The focus of this work will be on the supervised domain, nevertheless, an unsupervised clustering analysis is also explored. Multiple supervised classification algorithms are experimented, using a set of pre-classified documents, in order to comparatively evaluate their classification performances. Support Vector Machine (SVM), K-Nearest Neighbours, Multinomial Naive Bayes and Decision-Trees were used individually and, in order to seek to enhance the results, in conjunction with various techniques for pre-processing features. Latent Semantic Indexing (LSI), feature selection with different metrics and stemming were analysed.

Clustering with K-means, did not show underlying structure of the corpus which allowed text classification. SVM demonstrated to be the best classifier, achieving performances similar to the ones found in the literature. Feature selection with χ^2 method allowed to achieve classification scores close to the full feature set, using only 20% of it. LSI demonstrated to be powerful dimensionality reduction technique. With a reduction of 95% of the feature set, this method is able to reach, albeit marginally, better classification scores than the original full feature set. Stemming, on the contrary, shows only to be beneficial for corpus where the number of features is reduced. These pre-processing methods, even though did not presented performance increases as expected, shown to provide significant feature reduction, and consequently, decreased processing times while maintaining identical scores.

Keywords: Portuguese Legislation, Legal Texts, Text Categorization, Supervised Classification

Contents

List of Tables

List of Figures

Nomenclature

$\|C\|$	Number of categories in set C
$\|D\|$	Number of documents
$\|T\|$	Total number of terms/words in the feature space
C	Set of categories
c_i	Category i
D	Set of documents
d_i	Document i
N_i	Number of documents belonging to class c_i
N_t	Number of documents containing the word t
N_{it}	Number of documents where t and c_i co-occur
T	The vocabulary set
t	Term/word
t_{ij}	Term/word j in d_i
X	Document term matrix

Acronyms

BOW Bag of Words.

DT Decision Trees.

IG Information Gain.

IR Information Retrieval.

K-NN K-Nearest Neighbour.

LSI Latent Semantic Indexing.

MI Mutual Information.

ML Machine Learning.

MNB Multinomial Naive Bayes.

NLP Natural Language Processing.

POS Part-of-speech.

SVM Support Vector Machines.

TC Text Classification.

TFIDF Term Frequency / Inverse Document Frequency.

Chapter 1

Introduction

Portuguese laws are, similarly as in other legal systems, constructed following a structured manner. They are presented hierarchically by sections, which are by themselves, divided in articles. The sections, articles and other sub elements aim to encapsulate topics and context. The division of legislative text also follows an ordered numbering system, in order to permit to cite a particular element of the text. Moreover it allows to guide the reader to each topic and rule. Laws also reference and modify other laws in the same text. However, these documents are constructed in a normative and verbose manner, instead of an analytical and systematic structure. It is not uncommon when a new law is published to exist an uncertainty about the impact and coherency of it. Often laws require to be rectified or clarified. Moreover, due to the nature of all languages, when not carefully constructed it also allows subjective interpretations. This creates obstacles to the upkeep of legislation and to the knowledge of citizens. Legislation texts are crucial because they affect entire nations, and every citizen in them. Being so important, it should be easily accessible and understood, however due to its complexity and jargon, citizens seek specialized help.

Researchers and citizens are posed with a difficult problem when searching for topics of interest in legislation. In which documents lies the pertinent information? The analysis and narrowing of relevant information in large databases is not a trivial task, and can consume a lot of time. Moreover political and legislative documents could be of interest to researchers since in some way they reflect the social, economical and political situation of a country in a certain time frame. It is possible to observe trends of the political wing, reflections of economical influences and responses for new problematics of the world.

Nowadays most of the legislation is published freely by the governments in publicly available on-line databases, offering easily the data for researchers to work with and citizens to explore. The problem lies in the absence of systematic organization of the legal information, creating obstacles to its knowledge and access to both citizens and legal experts. Sophisticated search tools can help the user querying the system database, suggesting the most approximate results based on the keywords used for the search or even synonyms of it. However questions still arise from this approach. Are the words used enough to describe the subject and retrieve the most relevant information we are looking for ? Is the search result all that we can find about this particular subject in the database? It does not seem possible to guess

all the subjects a user might look for and all the topics which a document might contain, but we can try to narrow the search and fit the documents in predefined and concise topics. This may seem more restrictive and subjective than an open query system but it allows to consolidate the main concepts into an organized framework.

There is already some effort in creating a set of categories which represents the main topics of political agendas. The *Policy Agendas Project*[1] aims to create a coding scheme with 19 major topics and 225 specific subtopics. The main purpose is to use a generalized and organized code system to categorize, over time, the legislative content. A side project, *Comparative Agendas Project*[2], extends the initial project to other participating countries, being Portugal one of those, aiming for comparability among nations.

The main issue with classifying text into categories is that it requires a great amount of expertise, and consequently high resources. The quantity of already published legislative documents would require not only an extremely high amount of time but also a big effort of experts in the field. A research unit group from CIES (Centro de Investigação e Estudos de Sociologia) belonging to the ISCTE - University Institute of Lisbon started this task and provided a set of 1810 of already classified documents using the Policy Agendas Project code scheme. The goal of our work is to use the provided set of classified documents and explore the possibility of automatically classifying these documents into a set of predefined categories and analysing the possibility of extending it to the remaining documents in the Portuguese legislation which are still left to classify.

In computer sciences the area of Text Classification (TC) lies in a subset of Information Retrieval (IR), an area of extensive research giving the opportunity of exploring some already known successful methods and employing some new state of the art methodologies. When given a set of predefined categories the Text Classification goal is to automatically predict, given a document, its category. TC uses techniques from Machine Learning (ML) and IR. Last decades these areas received a lot of attention since the advent of the Internet due to the growing need of retrieving relevant information fast and accurately from large databases. Using these techniques from IR and ML the objective of this work is to show that there is a possibility of mitigating the necessity of extensive human labelling labour. By automating this process, it could allow researchers to dedicate their investigation time into other complex questions and aid citizens in the search of information in the legal turmoil. Moreover, it will be shown that legal text classification is practicable in reasonable computation time and with relative high accuracies.

1.1 Motivation

Portuguese legislation can be identified by number and date, type of legislation and source. It might not be simple and easy to find legislation relative to an specific problem and field, neither the identification fields allows easily to search for an specific topic or area of interest which could be useful for a common citizen. We can try to guess that certain topics are related with some sources, for example health

[1] Policy Agendas Project: `http://www.policyagendas.org/`
[2] Comparative Agendas Project: `http://www.comparativeagendas.info/`

can most probably be found in the Ministry of Health legislation, but that might not be entirely true for all ministries since responsibilities sometimes shift between mandates. Also, most of the time when searching for legislation we are not looking for broad topics, but instead specific and concise ones.

In Portugal, some websites dedicated to different fields of expertise, like education or civil and work rights, gather legislation specific to their area in a categorized fashion. Other private tools seem to provide services with some sort of grouping relevant to legislators, however no free and open tool or service was found that provided a similar purpose. Since nowadays a digital version of the Portuguese legislation is published on-line and it is easier to process computationally, allowing a more scientific and efficient exploration of the problem.

Data Mining, and Text Classification in legislation already rose the attention of social and political sciences researchers since it allows to observe trends, not only in the type of documentation that is produced, but also the intrinsic details that it deals with [1]. For example, it is possible to infer economic reactions by seeking for the number of documents published that deal with tax or inflation contents. Also, social indicators can be analysed by searching for keywords in published issues which address social rights, such as race or gender. Some researches went further and even tried to evaluate the political positions and decisions based on law contents [2]. This has a lot of interest for the analysis of how societies develop and change over time. Here we are looking for a general way of categorizing Portuguese legislation in an uniform way, which could also allow to study trends and other political issues by simply searching with dictionary keywords in the categories and analysing their distribution tendencies over time.

1.2 Portuguese Legislation System

The Portuguese legislation is published in *Diário da República* after being approved by the government council and the president of Portuguese Republic. *Diário da República Electrónico* (DRE) is the official website[3] for the portuguese government journal where business agreements, legal notices and proceedings of the government are published.

In DRE there are two types of publications, the *first* and *second* series. The first (1ª *Série*) covers the most important legislation issued such as the constitutional law and regular laws. The second (2ª *Série*) covers the normative documents issued by the Government, such as municipalities budgets or nominated personnel for administration positions. There are many different types of documents covered in the First Series Legislation (*Lei constitucional, Lei ordinária, Decreto, Portaria*, etc.), each with different purposes. It follows a structured hierarchy which depends on the type of the document, where the types in lower brackets must obey to the higher ones. The documents are also published by a ministry or a conjunction of ministries. The ministries can, in some way, be indicative of the subject which might be being discussed. For instance, the Health Ministry can publish laws relative to hospital financial credits, transfusion and transplant of organs, doctor statutes, etc. The document identifications are not intuitive nor the titles associated with them clear on the full subject which is being treated. Furthermore, when

[3]Diário da República Electrónico: `https://dre.pt/`

governments change, ministries can be modified and their names as well, meaning that over the years responsibilities and tasks can shift from one ministry to another.

Without prior knowledge on legal issues it is a complex task to find all the legislation relative to a particular subject. Moreover, for the reasons stated before, it is not easy for a citizen who does not follow legal matters, keeping updated, requiring most of the time professional help for simple needs. The categorization of legislation does not provide a full solution for these problems, but it can help citizens, researchers and even professionals. Categorizing with a computational algorithm could reduce the necessary documents that a researcher would have to look in to get information about certain topics.

The biggest problem with categorizing legal documents is the need for instructed coders to perform such task, which is very time intensive and many times expensive as well, and therefore the motivation of using ML for this process. One thing to bare in mind is that no automated classification system is able to provide 100% success, and such methods should be seen as way of amplifying the capacities of human categorization. For instance, if two coders are in discordance about a topic, a computer classification could aid in clarifying such situation. Or, it could also allow the expert to focus on the documents which were classified with less confidence and trusting in an automated classifier for the others. Moreover there are datasets which the content is easy to categorize whereas there are others where the task seems to be more complex, specially if the topics are tightly related. And finally, there is no ground consensus in a classification method, but rather methods that usually perform better with text. Due to the intrinsic characteristics of the content of a document collection or the language which is being classified, sometimes some methods perform better than others when in different datasets the opposite occurs.

1.3 Text Classification in Legal and Political Documents

Lately with the advance of Text Mining tools, researchers have started to centre their attention on new forms of analysing efficiently large collections of political texts. Grimmer and Stewart provides a very succinct introduction on how to address this problem, presenting the most common approaches of TC and content analysis [1]. Furthermore, he concludes that this kind of tools aims to offer help and guidance to researchers in analysing legislative matters, remarking that there is no substitute for careful and close reading of the collections.

The lack of scientific analysis of political and legal tools motivated already interesting works in the field. For instance, the exploration of the senate press releases to estimate the topics in the texts and the attention that political actors allocate to those estimated topics [3]. Instead of categorizing the text into topic categories, other kind of categorizations are also explored. The analysis of the political position of the parties in the context of social matters, left or right wing is one example [2, 4]. The most basic forms of text category assignment consists of creating dictionaries based on word analysis and assigning sets of words to classes, then classify based on the frequency of those words in the documents [5]. More sophisticated approaches can be used, for instance, by pre-classifying documents as a baseline to try to replicate the human coding, — the supervised approach — to classify the activism activity in the Russian

security forces [6]. Moreover, the classification of legislative texts, using the same principles, have been already explored with quite satisfactory results [7, 8]. No previous work with Portuguese legislation was found in the literature in respect to text classification. Here we will explore similar methodologies with data and category sets previously unexplored.

1.4 Collecting and Preparing the Documents

In order to run TC algorithms we require a plain text dataset of the Portuguese legislation. To this work it was only provided a database with the categories of the documents, not the documents themselves. Therefore, since the category records were identified by the respective publication dates, publication numbers and source, it was possible to identify and retrieve the respective documents from the DRE website. To retrieve the documents there were two options: either search and download the documents manually by the available search engine or, programming a script to download automatically these documents. Since the script allowed to collect extra information easily about the documents, it was the chosen approach. This information is often called *meta data*, and it is the available information in the web page about the document. The meta data contains information such as the publishing date, ministries involved, respective document identification plus a small summary of it.

The script which was used is called a *web crawler* since it crawls automatically through the website and collects information on the pages to a database. The documents are PDF files that are fetched in the same process of gathering the meta data. It was collected all the documents which were in between the dates of the oldest and youngest categorized documents, plus older ones which still allowed directly processing the text[4]. With this methodology a total of 187241 documents were collected from the website.

It was also necessary to normalize the identification numbers of the fetched documents to allow a correspondence between the given database of document categories and the database of collected documents texts.

1.5 Overview

The outline of this dissertation is as follows. In chapter 2 an overview of the basic concepts and methods of how to deal with textual data for the text classification problem is presented. In chapter 3, unsupervised clustering methods and the most common supervised classiers TC are presented in an introductory way. In chapter 4 is described the evaluation metrics that will be used to compare the methodologies and how to test the classifiers to present more statistically relevant results. In chapter 5 the experimental setup used is presented, describing the datasets used, the chosen methodologies and experimented algorithms. In chapter 6 the supervised and unsupervised experimentation results are presented and commented. Finally, in chapter 7 conclusions are drawn and future directions about the research are

[4]At some point in time, the uploaded texts to the DRE website did not contained ”selectable text”, meaning that the documents could probably be scanned and uploaded in PDF format.

outlined.

Chapter 2

Text Classification

There are two main approaches to TC, one is to directly encode an expert knowledge about the categories in the form of procedural rules. Another way is through ML, where a classifier is trained with labelled documents and then tries to predict the label for unseen documents into one or more classes. Since one of the major difficulties of the problem would be to manually classify by expertise, the later method is much more appellative since it is less costly and easier to keep.

2.1 Machine Learning Approach

If there is knowledge about the set of categories which one wants to label the documents into, then the task lies under the Supervised Text Classification domain. Supervised Classification consists in having a set of already labelled documents and training a classifier with (a subset of) them. If there is no knowledge about the document categories then it lies in the Unsupervised Classification domain. With unsupervised techniques it is not possible to guess which categories will emerge, and because of that, these techniques are, most of the time, only used to discover possible groups of classes. The supervised approach is the one which is commonly referred to as TC, and here the same convention will be used, unless stated otherwise. Since the main purpose is to categorize in a topic based fashion, the focus in this work will be on the supervised domain, while also an unsupervised evaluation, using a Clustering algorithm, will be performed to evaluate if the documents organize in a fashion similar to the given categories.

Formally given a set of documents $D = \{d_1, .., d_n\}$ and a predefined set of categories $C = \{c_1, .., c_m\}$, TC consists in the task of approximating an unknown category assignment function $F(d, c) : D \times C \rightarrow \{0, 1\}$. The classifier is the approximating function $\hat{F}$, where the goal is to build one as close as the F function [9].

In a text document, we can find different elements such as punctuation, words or numbering. The text can be also structured with titles, paragraphs or sections. Words can also be morphologically different, i.e. a noun or a verb, and also a sentence can be analysed syntactically. When that information is encoded to be interpreted by the classifier, a feature is created. A feature can represent a word, a set of

word characters or any other structure, which will be represented numerically to the classifier.

Depending on the application, a document could be classified into one, or more than one category. Multi-label classification is when a document can belong to multiple categories, and single-label classification when exactly one category is given do each document. There is also binary classification, a special case of single-label, when we classify the documents into belonging or not to a category.

The binary classification is quite relevant since it is the simplest form of categorization and also because of its broad usage. The single-label and also the multi-label problems can be simplified into a set of binary classifications, with equal to the number of classes, known also as the ”one-vs-all” approach.

2.2 Framework of a Text Categorization System

A TC system has many parts and variants depending on the purpose and techniques explored. The most common approaches and phases of a TC framework will be presented here. In the literature, there are different names for the same processes and often the concepts and purpose of each component end up blending.

A text categorization framework can be summarized essentially in three phases:

1. **Feature Engineering**: The text documents are preprocessed to be interpreted by the classifier. Here we identify, treat and filter the features with the purpose of maximizing the categorization success.

   (a) Feature Identification: Transforms the text into our features, which will be our representation of the documents. Involves identifying what will be considered as potential relevant information, such as the text structure or simply words, and filtering out elements of the text which we assume of few interest, like for instance, punctuation.

   (b) Feature Normalization: Features are processed to remove capitalized characters and, weighted according to their relevance in the collection. Occasionally algorithms are applied to reduce words to their base root.

   (c) Dimensionality Reduction: Since every word in the corpus is a potential feature, TC systems often deal with a very high number of unique features. There are several studied techniques which can help reduce the dimensionality of the data. Feature selection uses metrics to measure the features importance to the classifier and remove less relevant ones. There is also dimensionality reduction which consists in transforming the original set of features into a different dimensional space representation. The new space representation, with smaller dimension, is an approximation of the original data which aims to represent the most important information of the features.

2. **Classifier Training**: The training data, i.e. the previously labelled documents, are analysed in order to construct a classification model, which is used to predict a class label for an ”unseen” document. There are many classification algorithms used in data mining which have been shown

to be well suited for TC tasks as well. In this phase multiple experiments are performed in order to evaluate which of those better model our corpus characteristics.

3. **Classifier Testing and model evaluation**: The data unseen by the classifier, i.e. the data which was not used in the training phase, will be used to test the classifier. The results will be then compared against the expected labels and a formal evaluation of the classifiers and their accuracies will be performed.

In Unsupervised TC with document clustering, the *Feature Engineering* step is also followed, however, since no information about the labels is passed to the algorithm, it produces a set of clusters which does not allow the same kind of analysis as in the supervised counterpart. If the algorithm output is used for direct document classification then the resulting clusters would have to be manually identified and associated to categories. The *Classifier Training* and *Testing* phases would be substituted with a *Cluster Analysis* phase, which is usually the common approach in unsupervised techniques, where metrics are used to evaluate how the documents are organized.

2.3 Feature Identification and Document Representation

Human languages are very complex and encoding that complexity to identify the most important features (Feature Identification) is not an easy task. There are Part-of-speech (POS) taggers which allow to assign a tag to each word in the document [10, 11]. Each tag is, in practice, a classification in categories of nouns, verbs, adjectives, etc. It is a form of structure prediction which allows also to select for instance only nouns, while leaving out others which might be considered less relevant, like pronouns. This kind of tools are language dependent, meaning that often require to be trained, in a fashion similar to a supervised classifier, and also prone to errors.

Most TC tasks usually only consider individual words as features, ignoring punctuation and often numbers as well. However in some cases, for instance email spam classification, excessive punctuation is sometimes a good indicator of unsolicited email. Punctuation in TC for the legal domain does not seem relevant and numbers are often related with accounting which are highly variable and carry few class related information.

The process of transforming the document text stream into a set of possible features is often named *tokenization* and falls in the Pre-Processing phase, which will be detailed later [12].

There are different forms of representing a document, or in other words, identifying features:

- Bag of Words (BOW): This method consists in directly representing the document as a set of found words, where each word maps one feature, as an n-dimensional vector $d = \{t_1, ..., t_n\}$.
- Phrase based: It aims to keep, partially, the original structure of the text. One could index an entire phrase or subsets of it:
  - N-grams: A n-gram is a sequence of n-items in a text, where an item could be words or even characters [13]. For instance, a bigram (n-gram of size 2) of words of "The clean water"

consists in the combinations of {('The', 'clean') , ('clean', 'water') , ('the', 'water')}, being each of those our features. The former BOW is the 1-gram particular case of this representation.

  - Noun phrases: first an algorithm assigns POS tags such as nouns, verbs or adjectives to the individual words. It is possible to group each tag, or combinations of multiple tags, in order to capture the syntactic characteristics of the text.

- Semantic based: aims to capture the semantic concepts and relationships between words. Each word is linked to a representation of its meaning by indexing it an dictionary entry-like. It has a description of the word, or a set of words from a thesaurus which share the same meanings.

The way a feature can be represented computationally, may be trough a numerical vector, for instance, the number of occurrences each feature is found in a document. A particular case is the binary representation, where a feature is or is not present in the document. This method often do not provide beneficial results, and therefore in this work only vectors of frequencies of features will be considered [14, 15]. The feature vectors can be weighted in different ways, which will be discussed later in section 2.3.2.

When representing text as features there are some desirable characteristics which one should look for. A flat distribution of values of our features is preferable over too frequent words, since the latter usually carry few relevant information to the class. Some methods try to overcome this by weighting the feature vectors properly, which are going to be discussed in 2.3.2. Excessive number of (uninformative) features can jeopardize the classifier effectiveness and, also, its computational performance, problem which is known as *curse of dimensionality*. Common techniques to deal with it will be reviewed in section 2.4. Preferably, a feature set should also be noise free, which in the context of TC could be introduced by spelling errors or bad character recognition, for instance. Since we are dealing with very technical texts, and the source of our datasets come from the digitalized legislative text it is assumed that there are no errors of such kind.

2.3.1 Bag of Words

Phrase based representations often imply a bigger number of features, increasing greatly the dimensionality of the data. In the case of *noun phrases* representations, POS tagging is required and, as already stated, it requires adequate tools and algorithms. Also, several experiments have shown that sophisticated representations like those usually does not introduce significantly better results [16, 17]. Semantic based is an interesting concept and, for English languages, there is the *Wordnet*[1] database but does not seem to exist similar framework available for Portuguese. The most popular and straightforward approach among TC is the BOW representation. Its effectiveness has been widely demonstrated in many studies [18].

Due to the simplistic nature of this representation there is no consideration about the order of the words, meaning the structure of the document is also lost. Consequently it does not capture semantic

[1] Wordnet lexical database: `https://wordnet.princeton.edu/`

relationships between words. For instance, in ”Health Ministry”, both terms, when considered individually, lose its context meaning.

It is common to filter out words which are too frequent in the language since they tend to be statistically irrelevant. For instance ”the”, ”a” or ”but” can be found almost in every document and therefore they carry no value in distinguishing the documents from each other. These words are called *stop words*. It is not possible to have a predefined universal list, since depending on the subject of the corpus other words could be considered. To tackle the problem of potential irrelevant features, the weighing schemes become an important factor.

During the presentation of this work it will be always assumed the BOW model.

2.3.2 Feature Weighting

Since the feature space represents the words in the document it is of great significance to weight them according to their importance, as demonstrated in the experiments of Salton and Buckley: ”The main purpose of a term-weighting system is the enhancement of retrieval effectiveness” [14].

Term-Frequency/Inverse Document Frequency

One of the most popular feature weighting schemes in TC is the Term Frequency / Inverse Document Frequency (TFIDF). It is composed by the Term-Frequency (TF), which is simply the number of word occurrences, and the Inverse Document frequency (IDF), which is a measure of whether the term is common or rare across all documents, as proposed by [14]. Formally, the weight, w_{ij}, of the term t_i in the document d_j is:

$$w_{ij} = tf_{ij} \times \log(idf_i) = tf_{ij} \log \frac{|D|}{N_i} \tag{2.1}$$

where:

tf_{ij} is the term i frequency in document j,
$|D|$ is the total number of documents in the corpus,
N_i is the number of documents which contain the term i.

The term-frequency component (tf) alone cannot ensure acceptable discernibility between relevant documents. For instance, if a term is frequent in all documents of the corpus then all of them become relevant. To contradict this, the inverse document frequency (idf) factor computed by $\log |D|/N_i$ is introduced. The idf factor varies inversely with the number of documents in which a term appears, increasing the weight of terms which are present in few documents of the collection. Sometimes, researchers drop the $\log$ component but it is present to avoid favouring too much rare terms in the collection.

In a succinct way: TFIDF aims to favour terms which allow to better distinguish certain individual documents from the *corpus*, while penalizing the ones which are too frequent in the whole collection, like the previously referred stop words.

Scaling

Feature scaling or normalization is the process of standardizing the range of the features. Some algorithms are insensitive to scaling while others are not.

As previously said, the vector based representation of a document collection can be viewed as a set of vectors in a vector space, in which there is one axis for each term. If two documents discuss the same topic it is probable that both contain the same terms. However, if one of the documents is bigger than the other, then the relative frequencies might be identical in the two documents, but the absolute term frequencies of the bigger document will be larger.

Euclidean normalization (also called L^2 distance) is the most common norm applied to the feature space, converting the document vectors to unit length vectors. For each document d_j its scaled version d'_j is computed as:

$$d'_j = \frac{d_j}{||d_j||^2} = \frac{d_j}{\sqrt{\sum_{i=1}^{|T|} d_{ij}^2}} \tag{2.2}$$

where:

$|T|$ is the total number of terms in the document,

d_{ij} is the term i in the document j.

There are some drawbacks in Euclidean normalization. Longer documents can be verbose, meaning that the same content is repeated leading to only higher word frequencies. Other normalization methods such as *pivoted normalized document length* aim to compensate this by including the length in the normalized document representation [19]. However this is usually most relevant in IR systems and most of the works in TC which apply the Euclidean norm with a good feature weighting scheme show improvements.

2.4 Dimensionality Reduction

Due to the complexity of human languages, as the corpus grows, it is often expected to have a bigger dimension of the feature space. This happens specially when working in domain specific areas, where there is quite specific vocabulary. High dimensionality can increase drastically the computation time of some algorithms or even make them unusable. Moreover, many of the features are irrelevant (like the stop words), carrying few discriminating information, affecting negatively the accuracy of our classifier. Reducing dimensionality is a way of reducing *over-fitting* of our classifier, which happens when a classifier is tuned to the characteristics of the training data rather than the real characteristics of the categories. Furthermore, most of the times, dimensionality reduction can be applied without affecting significantly the performance of the classification [20].

There are two main approaches for producing a reduced set of features:

- Feature Selection: consisting in ranking the features by a specified metric and selecting a subset of the best ranked ones, eliminating all the others which do not achieve an adequate score.

- Feature Transformation: consisting in transforming features space into another space of lower dimension. This process reduces dimensionality by using smaller dimensions which might be linear or non-linear combinations of the original ones.

The most common techniques for reducing dimensionality of a dataset are stop-word removal, stemming and feature removal by its frequency. Stop-word removal, which is usually applied during feature identification, discards words *a priori* without any metric inference from the text. Stemming can be viewed as a process of feature transformation, and it is going to be detailed in section 2.4.2. Feature removal by its frequency follows an approach similar to the stop words, by removing features which either appear too frequently in the document collection or exactly the opposite, by removing the rarest ones. The later approach seems to be more counter intuitive, since one could be removing features which are very niche specific to a topic, allowing easily to classify them into it. The safest option is to discard simply the features which show up only once. How to discern about which features are safe to remove can sometimes be solved by employing techniques which will be described in the next section.

The feature selection methods presented here require the knowledge of the labels of the documents, and therefore it lies in the supervised domain approaches. While the feature transformation methods which are introduced later in this section, do not require any previous sort of labelling, allowing to be applied in the supervised and also in the unsupervised domain.

2.4.1 Feature Selection

The main objective of feature selection is to find a subset of features with performance similar to the full set of features. Feature selection is a form of simplifying the classification problem, which makes training and testing a classifier more efficient by decreasing the size of the data. Moreover, it can increase the classification performance by removing features which might mislead the classifier into erroneous classifications. If, for instance, the word ”violência” (violence) only shows in documents where we are talking about ”direitos” (rights), the classifier might not classify a document with the word ”violencia” in crime related laws. This is over-fitting introduced by the dataset, which can be potentially avoided by a careful selection of features [21].

The implementation can be viewed as a search problem based on a metric where a number of subsets are evaluated as candidates. Due to practical reasons, an optimal search is not possible most of the times. Instead, simplifications are made, and often it is only tested a predefined number of times, using subsets of the best scored features.

Sophisticated feature selection methods have been studied in Yang and Pedersen [22], Forman [20]. Here we present some of those:

Gini Index

The Gini Index, known as well in Decision Trees (DT) by Gini Impurity, and also used as Feature Selection metric, is a way of measuring the discrimination level of a feature. Being C the set of categories, the

Gini Index $G(t)$ for a given word t is defined by:

$$G(t) = \sum_{i=1}^{|C|} P(c_i|t)^2 \tag{2.3}$$

where:

$P(c_i|t)$ is the probability of the class c_i knowing that the term t belongs to it.

Higher values of $G(t)$ indicate a greater discriminative power of the word feature t. It indicates how a randomly chosen word would be incorrectly labelled if a randomly selected label from the class set was chosen. The main problem with the Gini Index is that, if a class distribution is skewed in the first place, then it may not measure accurately the discriminative power of the underlying features, since it will favour the ones that belong to the biggest classes. In order to better reflect the informative power of the features, expression 2.3 is usually computed with the normalized conditional probabilities with respect to the global probabilities:

$$P'(c_i|t) = \frac{P(c_i|t)/P(c_i)}{\sum_{j=1}^{|C|} P(c_j|t)/P(c_j)} \tag{2.4}$$

Information Gain

In Information Theory, entropy is the expected value of information contained in an event, sample or data stream. The more the randomness in the system, the bigger is the entropy in it. The rarer is an event, the greater will be the information it provides when it occurs. It is measured in *bits* and the more disordered is a system, the more bits are required to describe it. The general information entropy is given by:

$$H(x) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{2.5}$$

where:

$P(x_i)$ is the probability of a discrete random variable x_i.

Information Gain (IG) term, known also by *Kullback–Leibler* divergence, refers also to *average mutual information* as in Information Theory but here it will be described the same way Quinlan did in its work for *Decision Trees* and as Yang and Pedersen presents [23, 22].

In context of TC, IG measures the quantity of information obtained for category prediction by knowing the presence or absence of a term in a document. Being C the set of categories, the information gain $I(t)$ for a given word t is:

$$I(t) = H(C) - \sum_{x \in \{t, \bar{t}\}} P(x) H(C|x) \tag{2.6}$$

using equation 2.5 in 2.6 we get:

$$IG(t) = -\sum_{i=1}^{|C|} P(c_i) \log_2 P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log_2 P(c_i|t) + P(\bar{t}) \sum_{i=1}^{|C|} P(c_i|\bar{t}) \log_2 P(c_i|\bar{t}) \tag{2.7}$$

where:

$P(t)$ is the probability of the documents having the word t,

$P(\bar{t})$ is the probability of the documents not having the word t,

$P(c_i)$ is the probability of class c_i,

$|C|$ is the number of categories.

We can transform the equation 2.7 into words and classes frequencies:

$$IG(w) = -\sum_{i=1}^{|C|} \frac{N_i}{|D|} \log_2 \frac{N_{it}}{N_t} + \sum_{i=1}^{|C|} \frac{N_{it}}{N_t} \log_2 \frac{N_{it}}{N_t} + \sum_{i=1}^{|C|} \frac{|D| - N_{it}}{|D| - N_t} \log_2 \frac{|D| - N_{it}}{|D| - N_t} \tag{2.8}$$

where:

N_i is the number of documents belonging to class c_i,

N_t is the number of documents containing the word t,

N_{it} is the number of documents with word t belonging to class c_i,

$|D|$ is the number of documents in the corpus.

The greater the value of the IG of the word t, the greater will be its discriminative power. In the extensive comparative study performed in Forman [20], IG outperforms most of the presented feature selection algorithms.

Mutual Information

Technically known as *pointwise mutual information* but here it will be presented as in the work of Yang and Pedersen [22]. Mutual Information (MI) is a measure of the variables mutual dependence, which in this case will be the mutual dependence between features and classes. In other words, it measures how much knowing one of these, either feature or class, reduces uncertainty about the other. In this case it will be the information known for classifying a document into class c_i given that a certain feature belongs to that document. The MI between the word t and class c_i is given by:

$$MI(t, c_i) = \log \frac{P(t, c_i)}{P(t)P(c_i)} = \log \frac{P(t|c_i)}{P(t)} \approx \log \frac{N_{it} \times |D|}{(N_{it} + c_{i\bar{t}})(N_{it} + \bar{N}_{it})} \tag{2.9}$$

where:

N_{it} is the number of documents where t and c_i co-occur,

$N_{i\bar{t}}$ is the number of documents where c_i occurs without t,

$\bar{N}_{it}$ is the number of documents where t occurs without c_i,

$|D|$ is the number of documents in the corpus.

There are two common ways to determine the final score of a word t, by averaging or by choosing the maximum value:

$$MI_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) MI(t, c_i) \tag{2.10}$$

$$MI_{max}(t) = \max_{i=1}^{|C|} MI(t, c_i) \tag{2.11}$$

The first, by averaging, is the most popular in TC but in several studies often shows worse performances than other feature selection algorithms [24, 25, 22].

The two main differences between IG and MI is that IG contains information about the term absence and also normalizes the mutual information scores using the joint probabilities. The main problem with IG is that it is biased towards rare terms as we can see by the equivalent form: $MI(t, c_i) = \log \frac{P(t|c_i)}{P(t)} = \log P(t|c_i) - \log P(t)$. When words have similar conditional probabilities $P(t|c_i)$, it means that they are equally "common" in a class c_i. If the term is rare relative to the whole corps, meaning that it appears mostly in documents of the class c_i, then $\log P(t)$ will increase greatly, since when the probability $P(t)$ tends to zero, the term $\log P(t)$ will tend to $-\infty$. This effect does not allow comparing terms with great frequency differences [22].

χ^2 Statistic

The χ^2 statistic is used in statistics to test the independence of two events and the goodness of a fit. In TC it is a way of measuring the lack of independence between a word and a class. It has a natural value of zero if both are independent. It is compared to the χ^2 distribution with one degree of freedom.

It can be defined in terms of probabilities by:

$$\chi^2(t, c_i) = \frac{|D| \times [P(t, c_i)P(\bar{t}, \bar{c}_i) - P(t, \bar{c}_i)P(\bar{t}, c_i)]^2}{P(t)P(\bar{t})P(c_i)P(\bar{c}_i)} \tag{2.12}$$

and can be estimated as [22]:

$$\chi^2(t, c_i) = \frac{|D| \times (N_{it}\bar{N}_{i\bar{t}} - N_{i\bar{t}}\bar{N}_{it})}{(N_{it} + N_{i\bar{t}})(\bar{N}_{it} + \bar{N}_{i\bar{t}})(N_{it} + \bar{N}_{it})(N_{i\bar{t}} + \bar{N}_{i\bar{t}})} \tag{2.13}$$

where:

N_{it} is the number of times t and c_i co-occur,
$N_{i\bar{t}}$ is the number of times c_i occurs without t,
$\bar{N}_{it}$ is the number of times t occurs without c_i,
$\bar{N}_{i\bar{t}}$ is the number of times neither c_i nor t occurs,
$|D|$ is the number of documents in the corpus.

The weighted average or the maximum value can be combined to express the score respective to each term:

$$\chi^2_{avg}(t) = \sum_{i=1}^{|C|} P(c_i)\chi^2(t, c_i) \tag{2.14}$$

$$\chi^2_{max}(t) = \max_{i=1}^{|C|} \chi^2(t, c_i) \tag{2.15}$$

This method is fairly popular in TC and has also shown reasonable good results in the literature, whereas Yang and Pedersen as shown also that χ^2_{max} performs better than its averaged version [22, 26, 27, 28].

2.4.2 Feature Transformation

Feature Transformation (also called feature extraction) can be seen as a way of combining features instead of removing them, resulting in a new set of reduced dimensionality with different feature values. If the identified features are carefully chosen and weighted, it is expected that the new feature set, resulting from the transformation, contains the most relevant information in a reduced representation.

Stemming

Stemming is a technique which is often applied in the pre-processing phase. It consists in transforming inflected or derived words into their stem, base or root form. The stem produced by the algorithms are not normally identical to the morphological root of the word, since the stemmed words only need to map the same stem. This results in singular, plural and different tenses to be consolidated into a single stem, reducing considerably the dimension of the data. For instance, the words ”stemming”, ”stemmed” and ”stems” would all map to the word base ”stem”. Different stemming algorithms produce different results, but the purpose remains the same.

The stemming algorithms have to be tailored to particular languages. There are multiple approaches, such as trained statistic models, or by looking at pre-defined tables, but the most common approach is the *suffix stripping* method [29, 30] . Since languages follow certain rules, suffix stripping methods aim to cover those rules when de-constructing the words to their root form by hard-coding them. For instance, in Portuguese plural words usually end with an ”s”, so the stemming algorithm will simply remove that character. Naturally, languages have their exceptions and following the last example, the plural of ”cão” (dog) is not ”cãos” but ”cães” meaning that the stemmed ”cãe” would not map its base form. Usually stemming algorithms try to cover this sort of exceptions.

Latent Semantic Indexing (LSI)

When dealing with word features, without further analysis, each word is, as previously stated, assumed to be independent from each other. Latent Semantic Indexing (LSI), or Latent Semantic Analysis, tries to capture relations in the terms which are implied (latent semantics) [31]. The occurrence of some patterns of words gives a strong clue to the occurrence of others. The LSI technique mostly aims to favour Information Retrieval query systems, however, combining it in TC for dimensionality reduction was found to be quite useful. Furthermore, it addresses the problem of the use of synonymous, near-synonymous and polysemous in texts, which consists in multiple features (multiple dimensions) which could correspond only to a single feature in terms of meaning [31]. LSI aims to compress the document vectors information into vectors with a lower dimensionality by looking at patterns in the original data. Since the independence assumption is not true, this technique finds dependence among the features of the corpus. It then maps this newly found information into new set of vectors.

In the processing phase, the documents will represented in a document term matrix, where an example of one can be found in table 5.4. In this matrix, the lines correspond to the documents in the corpus, the columns to words, and each cell to the frequency (if no further preprocessing, such as weighting,

was applied) of the word in the corresponding document. LSI consists in applying Singular-Value Decomposition (SVD) to this document term matrix X. SVD results in the decomposition of X into three matrices, where their product reconstructs the original document term matrix:

$$X = T_0 S_0 D_0^T \tag{2.16}$$

where T_0 and D_0 are orthonormal columns and S_0 is a diagonal matrix of singular values. T_0 and D_0 are called left and right singular vectors respectively and S_0 is the diagonal matrix of singular values. The singular values of S_0 are ordered in decreasing magnitude.

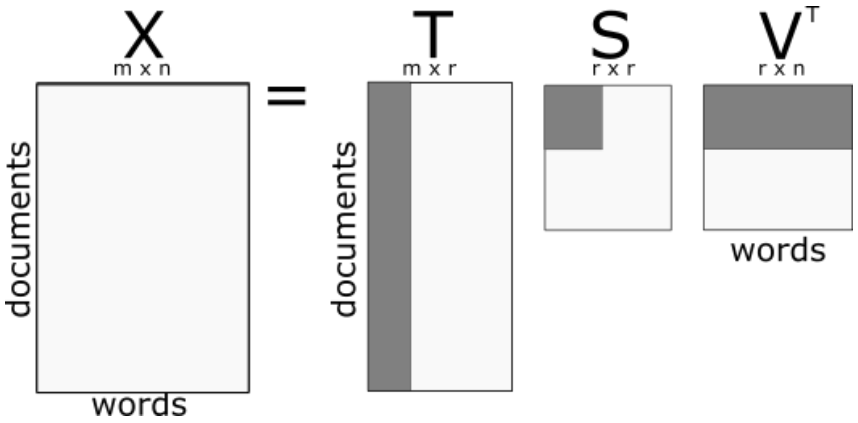


Figure 2.1: Singular Value Decomposition.

Since the singular values of S_0 are ordered, dimensionality reduction is performed by keeping the first *k* largest values and setting the smaller ones to zero. The process is represented in figure 2.1. The values matrices T and V^T are not exactly documents and words, respectively, but bases which allow to reconstruct the original matrix.

Bigger singular values are considered the most useful in reconstructing the original data. The product of the resulting matrices, which are simplified by deleting the columns and rows with zeros, is $\hat{X}$ which is approximately similar to X but of rank *k*. The shaded parts in figure 2.1 are an example of the dimensionality reduction of the data. The resulting model is therefore:

$$X \approx \hat{X} = TSD^T \tag{2.17}$$

A way of intuitively interpreting the SVD is to imagine it applied to an image. It we only select the first singular vector, we probably could distinguish some contours and shades, but as we select more components the image would get less blurry. By adding more singular vectors the image would get increasingly clear, however, the later singular vectors would contribute with much less information. With text, the intuition is the same: the first singular vectors of the decomposition capture most of the information about the latent semantics.

Chapter 3

Algorithms

3.1 Unsupervised Algorithm

There are two main unsupervised approaches commonly used in TC methods: topic modelling and clustering. Topic modelling, such as Latent Dirichlet allocation, use probabilistic models try to find and explain relationships in the data, associating the documents with a probability of belonging to a topic, which can be seen as a process of clustering with generative probabilistic models [32].

In unsupervised methods the data does not need to be labelled, allowing to be applied to any text data. The goal of unsupervised clustering is to cluster the documents without external intervention and additional knowledge besides its features. The resulting clusters should be in such way that the documents within a cluster are more similar than the documents belonging to different clusters.

3.1.1 K-Means Clustering

k-Means Clustering belongs to the set of partition clustering techniques, and is widely applied in many different areas of text mining and information retrieval [33].

The k-means algorithm aims to split the data into *k* clusters where each observation belongs to the cluster with the nearest mean, therefore its name. Each cluster mean is the mean of the documents d assigned to that cluster K, which are called centroids s. Each centroid is computed as:

$$s = \frac{\sum_{d \in K} d}{k} \tag{3.1}$$

The set of centroids $S = \{s_1, .., s_k\}$ are then used to compute the objective function which it is aimed to minimize, the squared error function:

$$min_s \sum_{j=1}^{k} \sum_{i=1}^{|D|} ||d_i - s_j||^2 \tag{3.2}$$

which is computed until it converges, meaning that the cluster assignments do not change, or until it reaches a maximum number of iterations.

There are variations of the implementations used to compute the k-means algorithm, two of the most popular are the Forgy's algorithm, and the Lloyd's algorithm [34, 35]. The later differs by using continuous geometric regions rather than discrete sets of points to define the clusters.

Usually *n* documents are randomly selected to form the initial mean centroids, meaning that the formed clusters vary with the initial selected documents. The standard algorithm, based on the Forgy's work, can be summed up by:

Algorithm 1 K-means Algorithm

Inputs:
Number of clusters k

Ensure:
Set of randomly selected n initial documents from the collection

1: **while** Cluster assignments change OR iterations $< n_{iter}$ **do**
2: Assign the documents to its closest centroid
3: After all document assignments, compute and update the new centroids: $s_i = \frac{1}{|S|} \sum_{d \in s_i} d$

One of the drawbacks of the K-means, since it is not guaranteed that the algorithm finds a global minimum, is that it becomes significantly sensitive to the randomly selected centres. To reduce this effect usually multiple trials are computed. Also, the chosen number of clusters can affect drastically the final distribution of the documents in the clusters.

3.2 Supervised Algorithms

Supervised algorithms require a set of labelled documents which are used to train and allow to provide a mapping function to the predefined class labels, based on the document features. Here are presented some of the most popular classifiers in the TC research.

3.2.1 Naive Bayes

It is a group of probabilistic models for classification based on the naive Bayes assumption. This probabilistic approach makes strong assumptions about how the data is generated: it assumes a mixture model for the generation of the documents where each class is a component of the mixture. Each mixture component provides the probability of sampling a particular term of its respective class. The term *naive* refers to the assumption where all features are independent of each other, given their respective class.

The probability of a certain document d_i to belong to the class c_k is given by:

$$P(c_k|d_i) = \frac{P(c_k)P(d_i|c_k)}{P(d_i)} \tag{3.3}$$

Since the probability of any document is the same and does not affect the relative value of the probabilities, $P(d_i)$ becomes only a scaling factor.

Based on how we represent the feature space, either taking into account the frequency of words or

only their presence in the document (binary representation), there are two main mixture models in Naive Bayes TC: *Multinomial Model*, *Multi-variate Bernoulli Model*, respectively.

In *Multi-variate Bernoulli Model* the document is represented in a binary way, either assuming 0 for absence or 1 for presence of the word in the document, represented here by B. The likelihood $P(d_i|c_k)$ is defined as:

$$P(d_i|c_k) = \prod_{j=1}^{|t|} B_{t_{ij}} P(t_{ij}|c_k) + (1 - B_{t_{ij}})(1 - P(t_{ij}|c_k)) \tag{3.4}$$

where $P(t_{ij}|c_i)$ is the probability of the word t_j in document d_i knowing it belongs to class c_k.

The probability of the word t_j, with respect to the whole vocabulary T, in class c_k in the Bernoulli model is given by:

$$P(t_j|c_k) = \frac{1 + \sum_{i=1}^{|D|} B_{t_{ij}} P(c_k|d_i)}{2 + \sum_{i=1}^{|D|} P(c_k|d_i)} \tag{3.5}$$

In the Multinomial Naive Bayes (MNB) Model the word frequency information is taken into account. It follows the bag of words representation, where the word position in the text is assumed to not affect its probability. Here we define N_{it} as the number of occurrences of the word t in the document d_i. The probability of a document given its class in the multinomial distribution is the following:

$$P(d_i|c_k) = (\sum_{j=1}^{|T|} N_{it})! \prod_{j=1}^{|T|} \frac{P(t_{ij}|c_k)^{N_{ij}}}{N_{ij}!} \tag{3.6}$$

The probability of a word t_j, with respect to the whole vocabulary T, which belongs to class c_k can be estimated by:

$$P(t_j|c_k) = \frac{1 + \sum_{i=1}^{|D|} N_{it} P(c_k|d_i)}{|T| + \sum_{s=1}^{|T|} \sum_{i=1}^{|D|} N_{is} P(c_k|d_i)} \tag{3.7}$$

Given a set of training documents it is possible to estimate the parameters $P(c_k)$ and, with one of the models, $P(d_i|c_k)$. The posteriori probability $P(c_k|d_i)$ of each class is estimated, by using equation 3.3 and selecting the class which corresponds to the highest probability:

$$\arg\max_{c_k} \frac{P(c_k)P(d_i|c_k)}{P(d_i)} \tag{3.8}$$

While the independence assumption of this model is not true in most of the real world cases, it often performs fairly well and, due to its simplicity, gained a lot of popularity in early TC research [36, 37, 38]. However, as mentioned by Sebastiani, usually more complex classifiers outperform the Bayesian approaches, yet this simple probabilistic methods are often used as benchmarks [21].

By incorporating the frequency of every word the multinomial model is expected to have better accuracy in datasets which have a large variance in the document length, as shown in the comparison of the two models by McCallum and Nigam [38].

3.2.2 Support Vector Machines

Support Vector Machines (SVM) are a kind of *Linear Classifier* which the output of the predictor is defined to be $p = \bar{W}_i \cdot \bar{X}_i + b$ where $\bar{X}_i$ is the word frequency vector from the document term matrix, and $\bar{W}$ is a vector of linear coefficients of the same dimensionality as the word vector, and b is a scalar.

In essence SVM attempts to linearly separate the document's different classes. The separator is commonly called *hyperplane*, as can be seen in figure 3.1, where one separates two linearly separable classes. The margin is defined as the sum of the distance of the closest training point belonging to each of the two classes. The points which are closer to the hyperplane allow to define our margin planes, which are the support vectors. It is possible to draw other separating hyperplanes and respective margins, based on different support vectors than the one represented in figure 3.1. However it is clear that there will be hyperplanes which will have lower margins. In test circumstances it is likely that an hyperplane with bigger margins will correctly classify more test instances than an one with shorter margins. If a point is close to the hyperplane, then there is almost 50% of chance of the classifier deciding either way. Therefore, the bigger the margin, the better will be generality of the classifier for unseen test examples.

Usually in real data sets the classes are not linearly separable. Although, it might be possible to separate most of the data into their correct classes with an appropriate hyperplane. The margin becomes softer, allowing some incorrect points to the wrong side of the plane, and because of that, this technique is commonly know as SVM with *soft margins*.

Support Vector Machines were introduced by Vapnik and the initial algorithm was later modified to allow soft margins, which is the most common version which is used today [39]. Later, Joachims proposed SVM in the context of text classification [40]. Known to be tailored for high dimensional problems, SVM only has to take into account a small part of the (training) data to construct the optimal hyperplane: the support vectors which define the margins. Joachims suggested that SVM was suitable to work with text classification because in corpora datasets it often has to deal with high-dimensional data. Nonetheless, document text data is quite sparse, since documents in big collections usually have few entries which are not zero. This means this data can be easily separable by those hyperplanes, and indeed his experiments did show that they are quite suited to the task [40]. Moreover with a *kernel trick* SVM can map the original non-linear feature space by using an inner product into a new linear space, where they can be separated linearly [41]. Here it will be only analysed in the linear context.

The separating hyperplane is described by the equation:

$$\bar{W} \cdot \bar{X} + b = 0 \tag{3.9}$$

and the two symmetric margin planes, which touch the support vectors, satisfy the following equations:

$$\bar{W} \cdot \bar{X} + b = 1 \tag{3.10}$$

$$\bar{W} \cdot \bar{X} + b = -1 \tag{3.11}$$

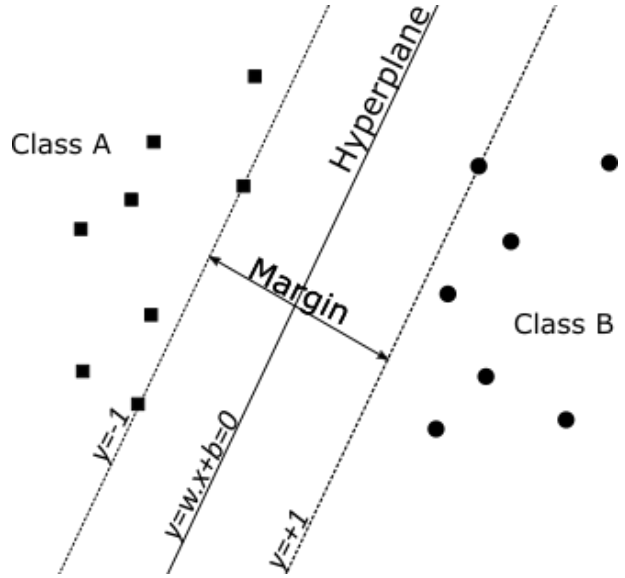


Figure 3.1: SVM Hyperplane and margins of two linearly separable classes.

Each row data point, $\bar{X}_i$, corresponds to the *i*th document vector, and y_i the respective class of the labelled training sample $\bar{X}_i$. The classes are said to be linearly separable if there exists $\bar{W}$ and b where the inequalities:

$$\bar{W} \cdot \bar{X}_i + b \geq 1 \quad \forall i: \quad y_i = 1 \tag{3.12}$$

$$\bar{W} \cdot \bar{X}_i + b \leq -1 \quad \forall i: \quad y_i = -1 \tag{3.13}$$

are valid for all the training set. The inequalities can be re-written as $y_i(\bar{W} \cdot \bar{X}_i + b) \geq 1$ and the vectors of X_i for which $y(\bar{W} \cdot \bar{X}_i + b) = 1$ are the *support vectors*. Sometimes the data is not linearly separated. As previously said, soft margins were suggested to allow small errors by introducing a parameter $\xi \geq 0$, which can be seen as a trade off between a large margin and a small error penalty. The separating hyperplanes with soft constraints may be expressed as:

$$\bar{W} \cdot \bar{X}_i + b \geq 1 - \xi_i \quad \forall i: \quad y_i = 1 \tag{3.14}$$

$$\bar{W} \cdot \bar{X}_i + b \leq -1 + -\xi_i \quad \forall i: \quad y_i = -1 \tag{3.15}$$

The function to minimize is therefore:

$$\begin{aligned} \text{Minimize} \quad & \phi = \frac{||\bar{W}||^2}{2} + C \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y(\bar{W} \cdot \bar{X}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{3.16}$$

The constant C regulates the importance of the margin and error penalty, whereas for large values

of C it acts as more conservative with hard-margins, and for small values it acts more permissive toward errors with soft-margins. The minimization of the functional 3.16, which is essentially finding the best separator hyperplane, is an convex quadratic optimization problem that can be solved using Lagrangian methods [42].

SVM are defined for two-class problems, meaning that for the multi-class problem it is handled in a different way. There are two approaches: the "one-vs-one" and the "one-vs-all" schemes. The first consists in constructing one classifier per pair of classes, resulting in $|C| \times (|C| - 1)/2$ classifiers. When testing with a new sample, if the classifier C_{ij} says the document is in class c_i it is added a vote for that class, otherwise a vote is added for class c_j. The class with most votes is selected. The "one-vs-all" method consists in training, as the name suggests, one class against all the others, resulting in one classifier for each class [43]. The test document is then run against all SVM classifiers, and the one which outputs the largest value is chosen. In Rifkin and Klautau [41] both approaches show very similar results, favouring the "one-vs-one" slightly for small datasets. In Hsu and Lin [44] the "one-vs-all" is the preferable method, indicating that the chosen approach does not have decisive impact in the results. However, in terms of computation time the "one-vs-all" approach is better since it requires training fewer classifiers.

SVM are one of the most popular classifiers in TC achieving really high performances compared to other classifiers, as shown in Dumais et al. [17], Yang and Liu [45]. It is a versatile algorithm which can be easily adapted to non-linearly separable datasets with the kernel trick with promising results, although the train computation times can be high when the dimensionality of the corpus is large as well.

3.2.3 Decision Trees

DT is a hierarchical decomposition of the training data in binary trees. The splitting of the data is performed at the nodes by a rule and, the same rule is applied to each child node recursively [23]. The decision rules are applied to the words of the documents. There are many different approaches with respect to the methodology of creating a DT, for instance ID3 and its successors C4.5 and C5. Here we will present other popular approach which is the Classification and Regression Tree (CART).

The goal is to decide which words of the training data, our documents, makes the best splitting based on their classes. The best splitter (word feature) is the one that decreases the diversity, i.e. increases homogeneity of the training samples by the greatest amount [46]. Homogeneity is translated into an impurity function for the node n. Let n_P be the parent node, n_L be the left child node, and n_R the right child node. Since the impurity of the parent node n_P is constant for any of the splits, the maximum homogeneity will be equivalent to the maximum change in the impurity function:

$$\Delta i = i(n_P) - P_L i(n_L) - P_R i(n_R) \tag{3.17}$$

where P_L and P_R are the probabilities of each respective child node.

There are different ways of measuring the best split, i.e. computing the impurity function Δi , and CART usually uses one of two metrics: conditional entropy of $P(c_i|n)$ given by the general entropy

equation 2.5 and the Gini Index impurity function:

$$i(n) = \sum_{k \neq l} P(c_k|n)P(c_l|n) \tag{3.18}$$

where k and l are the index of the class of the document and $P(c_k|n)$ the probability of class c_k provided we are in node n.

By applying the Gini Index impurity function 3.18 to the maximization problem maximization problem 3.17 we have:

$$\underset{t \leq t^{best}}{\arg\max} \left[-\sum_{k=1}^{|C|} P^2(c_k|n_P) + P_L \sum_{k=1}^{|C|} P^2(c_k|n_L) + P_R \sum_{k=1}^{|C|} P^2(c_k|n_R) \right] \tag{3.19}$$

The maximization problem corresponds to finding the best words t in the whole vocabulary T to associate to the splits in the decision tree. In the first iteration the impurity function is computed for each word in the document collection, then a binary search is performed to find the best split value using the decrease of the impurity as a measure of goodness. The word that gives the biggest decrease is chosen as the splitter for the node. This process is repeated into subsets of words until no split can be found that decreases the change of impurity in that node.

The nodes without child nodes in the tree are called leaf nodes, and to those a class is assigned. The chance of a document reaching a leaf and being incorrectly classified is called error rate.

In a simplistic manner, classifying a document corresponds to asking successive questions about the document along the nodes of the tree. Each node contains a *question* about the word of the document. For instance, is the word *dog* in the document ? Then it follows the left or right node based on the ”yes” or ”no” answer. Does the word *dog* appears more or less than three times ? And so on, until we reach a leaf, which will attribute the expected category.

The biggest problem with decision trees is that they tend to over-fit to the training sets. To avoid over-fitting and provide better prediction to more general cases, a process of pruning is performed, which basically consists in removing the branches which provide the least additional predictive power per leaf. Moreover traditional construction algorithms (C4.5 or the CART) are not designed to handle sparse data. However in some studies, DT shows competitive results compared to MNB [47, 48].

3.2.4 K-Nearest Neighbours

K-Nearest Neighbour (K-NN) is one of the simplest classification algorithms that has shown to perform well comparative with other popular classifiers [49, 48].

The algorithm is based on the similarity shared between the documents and their labels. In the training phase, K-NN simply stores the multidimensional feature vectors with their respective labels, without having actual ”learning” or internal parameters estimation. For the test phase it considers, according to a distance metric, the *k* nearest training examples which are closest. The major class among the *k* nearest samples is chosen as the class for the test document.

The *k* neighbour number is a predefined parameter of the algorithm. If the value of *k* is too small then the results may become sensitive to the noise in data. When a dataset is skewed, meaning there are classes which have a larger number of training examples than others, a big value of *k* might include too many (distant) examples of only one class and therefore losing sensitivity from the local data classes.

There are many distance metrics variations which are able to achieve more effective classification results depending on the kind of data which is being classified [50]. To counter the effect of imbalance in the data weighted approaches were developed, where either the class or features imbalances are taken into account [51, 52]. Others use the inverse distance from the test document to weight accordingly, resulting in a bigger influence from the closest neighbours than the ones which are further away [53]. The most common distance metrics are variations of the *Minkowski* distance, such as *Euclidean* distance when $p = 2$ or the *City block* metric when $p = 1$:

$$D(\bar{x}, \bar{y}) = \left(\sum_{i=1}^{|t|} |\bar{x}_i - \bar{y}_i|^p \right)^{1/p} \tag{3.20}$$

where x and y are document vectors, for which the distance is computed.

The advantage of the K-NN is that the cost of the learning process is zero, remaining all the computation cost to the prediction. Due to this, when the dimensionality of the corpus is large, it can be computationally expensive. Yang and Liu [45] show that, even though K-NN is a simple classifier, it can achieve performances comparable to the SVM for some corpus.

3.2.5 Meta Algorithms

Meta algorithms refer to the classification strategies that combine multiple classification algorithms aiming to increase the overall accuracy. These combinations can be performed by the same model on different subsets of the training data, the *boosting methods*, or by combining results from different classification algorithms, the *ensemble methods*.

Ensemble Methods

Ensemble methods, also known as *stacking*, are based on the assumption that multiple classifiers may perform better than a single one when their results are appropriately combined. The main motivation for these approaches can be seen in multiple studies, where a classifier outperforms all others in a specific problem or subset of the data but not in the overall problem. Moreover, if a particular classifier fails to infer correctly the categories, the overall system might be able to still recover from the error [54].

If ensemble classifiers are accurate and diverse then we have a sufficient condition for their performances to be better than their individual counterparts [55]. An accurate classifier is one that has an error rate better than random guessing, and a pair of classifiers are diverse if they make different errors on new data. The main motivation here is that the probability of the majority of the classifiers of making the same mistakes is low, assuming the errors are uncorrelated. Simply by vote majority, the correct classification should prevail over the sporadic errors.

Many ensemble techniques can be found in the TC literature with interesting results. Simple weighted linear combinations of the obtained scores or by simple vote majority decisions have shown to work [56, 57, 58]. Lam and Lai proposed a method that can estimate a suitable classification algorithm for each class in the data, based on their specific statistical characteristics, where a final system could end up with multiple classifiers for different classes [59]. Moreover, Bennett et al. presented also a more empirical way of combining classifiers using reliability variables of the text to choose dynamically which classifiers might perform best under those characteristics [60].

Boosting and Bagging Methods

Usually the boosting and bagging approaches are used in conjunction. Instead of using different classifier methods, *boosting methods* use the same method to train multiple parts of the training data, creating different models for each part. In TC the boosting technique was introduced by Schapire et al. [61], one of the most popular boosting algorithms named *Adaboost*, and in Schapire and Singer [62] presented other variations of the original algorithm, which have shown very promising results. Normally boosting methods are applied to what are usually called week learners such as Naive Bayes [63, 64] and Decision Trees [55, 65].

The main objective of the boosting method is to find a better classification model by combining many weak learning classifiers. It is assumed that these classifiers generate *weak hypotheses* for the classified data, where each may be only reasonably accurate. The boosting algorithm runs this weak classifiers in multiple sets of trials, and then combines them into a single rule called the final hypothesis. The major difference of this method is that instead of training the classifiers in a parallel and independent way, the classifiers are trained sequentially. After training the i classifier, the $(i+1)$ classifier is constructed on the training data where the classifier before was not able to correctly classify. The documents are weighted accordingly to the difficulty of being correctly assigned to their class, so that the next classifiers can iteratively focus on correctly solve the ones with bigger weights. In the end, a weighted linear combination of the classifiers is used to present the final decisions.

The bagging methods are generally designed to reduce over-fitting of the learning process [66]. This algorithms use samples with replacements from the underlying original data of the corpus to train the classifiers. The results of these samples are then combined in to yield the final result. These techniques are specially effective when small changes in the data leads to unstable classification algorithms like noisy data or classifiers which tend to over-fit, such as Decision Trees or boosting techniques [67].

3.3 Semi-Supervised Algorithms

Labelled data is hard to obtain, while unlabelled data is frequently easier to collect. Following this motivation, Semi-Supervised algorithms aim to address this problem by using the unlabelled data together with the labelled data to produce robust classifiers.

The semi-supervised algorithms are generally distinguished by inductive learning and transductive learning. Inductive learning refers to the methods which are only concerned in creating a global model

for the whole data being it labelled or unlabelled. Transductive learning does not handle unlabelled data and therefore needs to infer the categories of the unlabelled data before performing the classification based on their similarity to the already labelled data.

There are several approaches in semi-supervised learning: Self-Learning, co-training [68], transductive SVM's [69], Expectation Maximization with generative mixture models [70] and graph-based methods. When the data is well clustered, Expectation Maximization methods are a good choice, whereas if the feature set can be split clearly into two different sets, then co-training should be considered.

3.3.1 Self-Learning

The Self-Learning semi-supervised approach is one of the simplest and commonly used methods. It consists in training a classifier, such as the ones previously presented, with first only labelled data. Then, the classifier is used to classify the unlabelled data. Using the unlabelled data predictions plus the labelled ones the classifier is re-trained and the process is repeated, using its own predictions to train itself.

The advantage of this method is the possibility of wrapping any complex classifier easily. However mistakes produced by earlier classifications reinforce themselves. A form of mitigating this problem is to use only the most confident predictions to re-train the class, or dropping the prediction of a document if the confidence falls bellow a certain threshold.

3.3.2 Graph Based Methods

Graph-based semi-supervised methods make the assumption of consistency of the data, which means that points close to each other are more likely to share a label, and consequently if the data is structured by forming clusters, that points in the same structure are likely to share a label as well.

It can be viewed as estimating a function F for the graph, where the function should map as closely possible the given labelled data and be smooth in the whole graphed data. This process can be expressed in a regularization framework, where the mapping is achieved with a cost function, and the smoothness with a regularizer. Several graph-methods have been proposed, each exploring different ways of the assumption of consistency by varying the loss function and regularizer, such as Mincuts [71], Markov Random Fields [72], Manifold Regularization [73] and many others. An extensive survey on such approaches can be found in [74].

Here it is presented the Local and Global Consistency method proposed by [75]. For a document set $D = \{d_1, ..., d_l, d_{l+1}, ..., d_n\}$ and a category set $C = \{1, ..., c\}$ where the first $d_i (i \leq l)$ are the labelled documents, and the remaining $d_u (l + 1 \leq u \leq n)$ the unlabelled documents. Let Γ denote the set of $n \times c$ matrices with non negative entries. A matrix $F = [F_1^T, ...F_n^T]^T \in \Gamma$ corresponds to the estimated classifications of the documents, where F can be seen as a vectorial function which assigns a vector F_i to each document d_i. Let $Y \in \Gamma$ be another $n \times c$ matrix where each line is a vector with 1's in the columns corresponding to the classes for which the respective document is labelled. The resumed process can be found in Algorithm 2.

Algorithm 2 Local and Global Consistency Algorithm

Inputs:

Input parameter $\alpha \in [0, 1]$

1: Compute the affinity matrix W where $W_{ij} = exp(-||d_i - d_j)||^2/2\sigma^2)$

2: Construct the matrix $S = \mathcal{D}^{-1/2}W\mathcal{D}^{-1/2}$ where $\mathcal{D}$ is a diagonal matrix with (i, i) elements equal to the sum of the i-th row of W.

3: Iterate $F(t + 1) = \alpha SF(t) + (1 - \alpha)Y$ until convergence.

4: Being F^* the last iteration, label each document d_i as $y_i = \arg max_{j \leq c} F^*_{ij}$

The algorithm can be seen as a graph $G = (V, E)$ construction where the set of documents D are the vertices, and the edges are weighted by W. The parameter α specifies the relative amount of information retrieved from neighbours and initial labels. The second step is the symmetrical normalization of the weights. The third step consists in passing to the documents vertices the neighbours information while retaining partially the information of the initial labels which is regulated by the parameter α, corresponding respectively by the first and second terms.

The cost function associated with F for the iteration (3) in Algorithm 2 is defined as:

$$\mathcal{Q}(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij}\left|\left|\frac{1}{\sqrt{\mathcal{D}_{ii}}}F_i - \frac{1}{\sqrt{\mathcal{D}_{jj}}}F_j\right|\right|^2 + \mu\sum_{i=1}^{n}||F_i - Y_i||^2\right) \tag{3.21}$$

Where $\mu > 0$ is a regularization parameter. The classifying function is given by:

$$F^* = \arg\min_{F \in \Gamma} \mathcal{Q}(F) \tag{3.22}$$

To achieve successful classifications in semi-supervised problems the classifying function has to be sufficiently smooth with respect to the structure labelled and unlabelled data. Therefore, the local and global consistency assumption of the data. Zhou et al. showed in its study that the proposed method, based on the same assumption, was effective in text categorization [75].

Chapter 4

Evaluation Metrics and Validation Methods

In a supervised TC classification experiment it is necessary to have two sets of documents. One is the training set which is used for estimating a classifier function, and the other is the test set, where its results are used to evaluate the performance of the classifier.

Both document sets should be mutually exclusive, meaning that the test set should not be used in the process of training and vice-versa, otherwise biased results could be obtained. Usually, the training set is bigger than test set, since more information is required to train the classifier. There are several metrics proposed in the literature to evaluate the classification performance, each one with its advantages and drawbacks.

In unsupervised clustering, evaluating the performance is not as simple as comparing the predicted label outputs of the methods. Since it is unsupervised, there are no real classes given to the documents. Therefore, evaluation metrics for unsupervised methods, instead of matching predicted labels to known ones, have to take into account the separation of the clusters. The greater the similarity of the unsupervised separation to the real classes, the better will be the unsupervised method.

Evaluating the performance of a classifier should be measured not only in terms of effectiveness but also its execution time.

4.1 Classification Scores for Unsupervised Methods

There are two kinds of clustering evaluation: internal and external. Since usually there are no labelled documents in unsupervised methods, most of the evaluation to infer the quality of the algorithms have to rely on characteristics of the data that was clustered. This is called internal evaluation. When there is knowledge about the true labels of the documents, even though it is not used in the process, the quality of the clustering classification can be evaluated with it. The use of previously known information about the data is called external evaluation.

In general, internal clustering evaluation combines two types of measurements. It can be related to

the closeness of the cluster elements, *compactness*, or how distinct two clusters are by measuring their distances relative to each other, *separability*.

4.1.1 Silhouette Score

The Silhouette score is an evaluation metric which aims to unify these two concepts [76]. For each document, or sample, in the dataset it combines the distance between samples in the cluster (compactness) and the distance to the closest clusters where the samples belong (separability).

For a given cluster k_j the silhouette measure of each sample i in it is defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{4.1}$$

Which can be also be written as:

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases} \tag{4.2}$$

Where:

$a(i)$ is the average distance between the *ith* sample and all the samples in cluster k_j,

$b(i)$ is the minimum distance between sample i in cluster k_j and all the samples in the nearest cluster.

The final silhouette score is simply the mean of all document silhouettes.

The best scores of the silhouette tend to 1 and the worst to -1. Therefore, the smallest the mean distance $a(i)$ the more similar are the documents in the cluster, meaning well matched samples. Whereas the bigger is the $b(i)$ the more separated are to the closest cluster, implying that the clusters are well defined. Values close to 0 indicate overlapping clusters.

As already stated, when the truth labels of the documents are known, external validation metrics can be used. The output labels of unsupervised algorithms are unpredictable, meaning that the same subset of documents, which were labelled into the same cluster in a previous trial, can have a different one in the following experiment. External metrics are often computed based on the combinations of documents belonging to the same group, which are then compared with the groups of the known labelled documents, instead of just matching predicted labels. One of this metrics is the V-measure score.

4.1.2 V-Measure

V-measure score is an entropy based metric which joins two criteria for cluster evaluation [77]. Homogeneity evaluates how pure is a cluster relative to the labels of the documents which belong to it. A cluster method satisfies homogeneity if each of the clusters contains only documents of the same class. The other measure, completeness, evaluates how documents of the same class are aggregated together. A clustering method is said to satisfy completeness if all the documents of the same class belong to the same cluster.

To evaluate how homogeneous the clusters are, the conditional entropy of the class distribution $H(C|K)$ is computed as in equation 2.5, where in the perfect homogeneous situation, with all the documents of the same class assigned to the same cluster, the entropy will be zero since there is no disorder. Since it is dependent on the size of the elements belonging to the class, to avoid the skewed class problems it is normalized by $H(C)$. When the clustering provides no new information then $H(C|K)$ is at its maximum value and equal to $H(C)$.

For a set of classes $C = \{c_1, .., c_j\}$ and a set of clusters $K = \{k_1, .., k_j\}$ the entropies are computed by:

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{n_{ck}}{|D|}\log\frac{n_{ck}}{\sum_{c=1}^{|C|}n_{ck}}$$

and

$$H(C) = -\sum_{c=1}^{|C|}\frac{\sum_{k=1}^{|K|}n_{ck}}{|D|}\log\frac{\sum_{k=1}^{|K|}n_{ck}}{|D|}$$

where n_{ck} is the number of documents belonging to class c and to cluster k, and $|D|$, the total number of documents.

And homogeneity is defined as :

$$homogeneity = \begin{cases} 1, & \text{if } H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)}, & \text{otherwise} \end{cases} \tag{4.3}$$

The *completeness* measure is computed similarly to *homogeneity* since each other are symmetrical. When there is maximum completeness the entropy $H(K|C)$ will be zero as well. The entropies $H(K|C)$ and $H(K)$ for the completeness case are defined as:

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{ck}}{|D|}\log\frac{n_{ck}}{\sum_{k=1}^{|K|}n_{ck}}$$

and

$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}n_{ck}}{|D|}\log\frac{\sum_{c=1}^{|C|}n_{ck}}{|D|}$$

Completeness can now be defined as:

$$completeness = \begin{cases} 1, & \text{if } H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)}, & \text{otherwise} \end{cases} \tag{4.4}$$

V-measure is the weighted harmonic mean of the above two criterion:

$$V_\beta = (1+\beta^2)\frac{homogeneity \cdot completeness}{\beta^2 homogeneity + completeness} \tag{4.5}$$

When homogeneity and completeness are weighted equally, $\beta = 1$:

$$V_\beta = 2\frac{homogeneity \cdot completeness}{homogeneity + completeness} \tag{4.6}$$

The V-measure is actually equivalent to the Mutual Information but normalized by the sum of the class entropies.

4.2 Classification Scores for Supervised Methods

In TC, the main objective of the classifier is to fit all the test documents in their correct categories. However, finding only how many were correctly classified does not tell us how the system behaved when it incorrectly guessed a classification. Different metrics usually cover different aspects of the behaviour of our classifier. Most of the classification metrics used in Natural Language Processing (NLP) come from the classical IR, such as *recall* and *precision*. After a metric is defined the classifier parameters can be tuned to achieve the best results possible by performing multiple tests experimentally.

The outcomes of an experiment prediction for a test document can be found in table 4.1 similar as the one found in [78]. With respect to a class c_i:

Table 4.1: Contingency table.

Category c_i		expert classification	
		YES	**NO**
classifier decision	**YES**	TP_i	FP_i
	NO	FN_i	TN_i

TP_i are the documents correctly labelled into c_i.

FP_i are the documents incorrectly labelled to c_i which belong to another class.

FN_i are the documents which belong to class to c_i but were not labelled as such.

TN_i are the documents which do not belong to class c_i and correctly were not labelled as such.

We can now define:

- *Recall* is the number of correctly labelled documents divided by the number of elements that actually belong to the class. The recall with respect to class c_i is:

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{4.7}$$

- *Precision* is the number of correctly labelled documents divided by the number of elements that were labelled as belonging to the class. Formally, the precision with respect to class c_i is:

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{4.8}$$

4.2.1 F-Score

As Lewis [78] pointed out, a classifier can achieve very high recall by rarely deciding NO, or very high precision by rarely deciding YES. Therefore both recall and precision should be used in evaluating the effectiveness of a model. In the case of TC recall and precision are not independent of each other because if a document has been classified under the wrong category (decreasing precision) this also means it was not classified in the right category (decreasing recall).

A popular metric among TC is the F-score. It takes into account both precision and recall, and can be interpreted as a weighted average of both. Generally:

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 precision + recall} \tag{4.9}$$

Here we will only consider the F1-score, where precision and recall are weighted equally. Formally:

$$F1 = 2\frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FN + FP} \tag{4.10}$$

Usually there is an inverse relationship between recall and precision. When tuning the classifier to try to improve one of the scores, the other reduces. This is the main motivation for using combined measures like the F-score. F1-score is robust when trying to maximize both precision and recall, however if one of those is much lower than the other it affects the score greatly while giving no indication on which is responsible.

4.2.2 Averaging the Scores

How we compute the global scores affects the perception of effectiveness of our system. There are two different averaging approaches, micro-averaging and macro-averaging.

Micro-averaging is computing the score over all individual decisions of every class. It can be expressed formally:

$$precision_m = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)} \tag{4.11}$$

$$recall_m = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)} \tag{4.12}$$

Macro-averaging is computing the score for each class individually, then averaging equally every

class:

$$precision_M = \frac{\sum_{i=1}^{|C|} precision_i}{|C|} \tag{4.13}$$

$$recall_M = \frac{\sum_{i=1}^{|C|} recall_i}{|C|} \tag{4.14}$$

The F1 averaged scores are then computed with equation 4.10 where F1 micro-averaged score is denoted by $F1_m$ and the F1 macro-averaged score by $F1_M$.

In micro-averaging every document counts the same in the final average, while in macro-averaging each class counts the same. This is quite relevant because it can lead to very different results. When classes are very unbalanced, meaning the number of documents in each class is considerably different, the class with the most documents might dominate the classification process. The classes which have fewer training documents will provide less information, and consequently, less generalization power to the classifier. Because of that, results in such categories tend to be lower and, when macro-averaging, this effect will be more pronounced than when micro-averaging.

4.3 Cross Validation

Cross validation is a technique aiming to assess how the results of the classifier will generalize with data which it has not seen yet. A way of assessing the classifier is to split the (classified) data in two sets, as previously stated, the training set and the test set. Then multiple trials are performed to reduce variability, each with different partitions, and the results are averaged to present the final result.

In the parameter tuning phase we aim to fit the model as much to the train data, thus the main purpose of cross-validation is to avoid the known over-fitting effect, which is likely to happen on small datasets. The cost of performing this approach is that our training data will now be smaller, and our predictions will be less accurate. Therefore we can say that the results presented will be roughly the worst estimate.

K-fold cross validation consists in splitting randomly the dataset into *k* mutually exclusive subsets, called the folds, of approximately equal size. The classifier is then trained and tested k times, where at each turn one set is used for testing and the rest used in training.

Random sub-sampling consists in randomly splitting the data into training and test sets with fixed sizes. The main assumption that is violated in random sub-sampling is the independence of instances in the test set from those in the training set, meaning that, some samples may be selected more than once and whereas others may never be selected. The main advantage of the random sub-sampling in relation to the k-fold method is that the proportion of the splits are not dependent on the number of iterations, being useful when the dataset is small.

As already stated, repeating provides a better estimate to a complete cross-validation. This would involve testing and training with all possibilities, which is not normally feasible. For both methods exists *stratified* variants where the splits contain approximately the same proportions of the labelled documents

as the original dataset. An extensive explanation of these methods can be found in [79].

Chapter 5

Experimental Setup

5.1 Datasets

Corpus, or *dataset*, is a set of documents which are used as data in TC. A dataset in TC consists in a collection of documents where, depending on its purposes, may have classified documents for Supervised Classification, no classified documents for Unsupervised Classification or a mix of both for Semi-Supervised Classification. This chapter presents some statistics of the corpus to describe the data used in this work.

5.1.1 Diário da República (DRE) Corpus

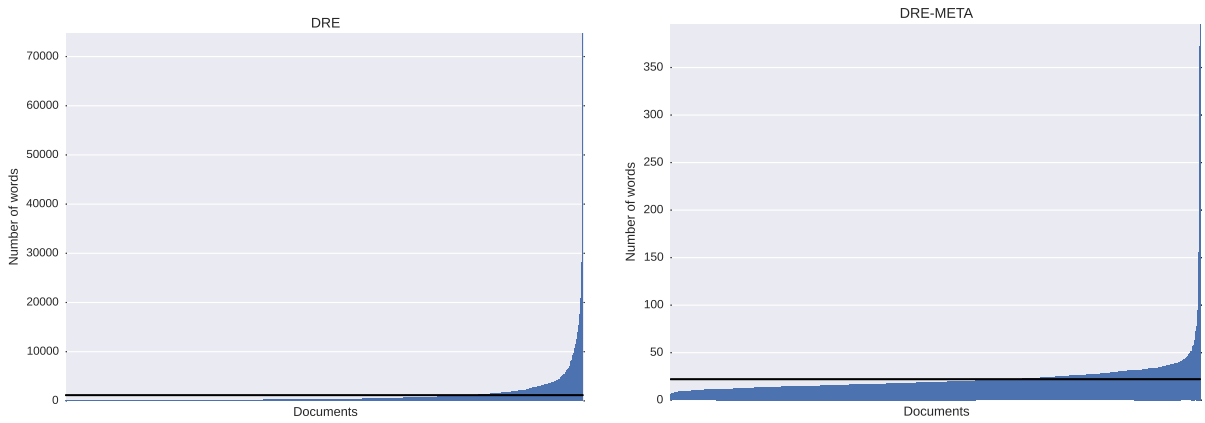
The text collection consists in 1810 classified documents and their texts were collected using a *web scrapper*. The classified documents used here only cover the first series, which is the most important one. Using the whole integral text of the documents a dataset was created which will be called simply as *DRE* dataset.

From table 5.1 it can be seen that there is quite lexical diversity, showing that even for a small set of documents the number of features is quite high. The number of words will be equivalent to our number of features, due to the BOW approach. Also, the statistics here are based on the already processed corpus, leaving out for instance stop words as it will be described in section 5.2.

Table 5.1: DRE corpora statistics.

Corpus	Total Words	Unique Words	Mean words per doc
DRE	2104896	36870	1162.926
DRE-META	40019	4851	22.110
DRE-GAP	830799	19320	1042.408

The major categories and the distribution of the documents in each can be observed in table 5.2 and the minor categories and its respective distribution in table A.1. At first glance it is easily observable that there is a great class imbalance. Moreover there is also a great unbalance in the document lengths which is observable in figure 5.1 meaning that there are documents which have a feature space much larger than the others.



(a) DRE (b) DRE-META

Figure 5.1: Document length distributions. The plotted black line corresponds to the mean document length.

Table 5.2: Major Categories and document frequencies for the DRE and DRE-META corpus.

Categories	**NºDocs**
Macroeconomia	105
Direitos e Liberdades civis e das minorias	14
Saúde	41
Agricultura, Pecuária e Pescas	26
Trabalho e emprego	16
Educação	140
Ambiente	12
Energia	10
Transportes	146
Justiça e Direito	112
Políticas Sociais	30
Habitação	87
Sector financeiro, indústria e comercio	86
Defesa	49
Ciência, tecnologia e comunicações	33
Comércio externo	11
Politica externa	50
Governo e administração pública	831
Recursos naturais e Património	11

5.1.2 DRE-META Corpus

An advantage of collecting text information using a scrapper is that the website, besides the integral text of the document, also offers a small summary and respective meta-data, such as the ministry that emitted it, date, and its type which can be used in classification as well. For experimentation a dataset with the source (ministry that emitted it) plus the summary, was created with a label set equal to the *DRE* corpus, which can be found in table 5.2. This dataset will be referred to as *DRE-META* dataset

In table 5.2 it can be seen that the number of unique words is significantly lower than the DRE corpus, leading to a quite small dimensional feature space. The document length distribution, which can be seen in figure 5.1, in comparison with the DRE corpus, is more evenly distributed, which means we are processing documents with reasonable equal feature quantities.

5.1.3 DRE-GAP Corpus

The most populated class in the *DRE* corpus represents approximately 46% of the whole *DRE* corpus. Since it represents a big chunk of the data, a subset with only the ”Governo e administação pública” category was created. To this corpus subset it will be referred to as *DRE-GAP* corpus. In this sub-dataset, the minor categories will be used as the major categories of the documents.

Since in the minor categories of the documents belong to a major broader topic, it would be expected that the features of the documents would be also similar between each others, even though it should have vocabulary specific to that minor area. The goal here is to evaluate the possibility of a TC system being able to categorize the documents into such specific minor categories. Unfortunately, the labelling distribution of the minor categories is also quite skewed, and some sub-classes have very few documents. Such classes, with lower than 10 documents, will be removed from classification tests, because in a such skewed set it is expected that those classes do not have enough discernibility power to do a proper classification and will only harm the overall performances. The final set of classes and respective document frequencies can be found in table 5.3. The document length distribution is not plotted because it is very similar to the DRE corpus, which is not surprising since it is a big subset of it.

Table 5.3: Major Categories and document frequencies of the DRE-GAP corpus.

Categories	NºDocs
Eficiência governativa	339
Administração Pública	197
Nomeações e exonerações	35
Condecorações, reconhecimento público, fabrico e cunhagem de notas e moedas	119
Compras Públicas	46
Gestão do setor público	21
Administração fiscal	25
Transição e consolidação da democracia	15

5.2 Preprocessing

The preprocessing phase consists in treating the text collection in such way it can be properly interpreted by the classifier, when constructing the features. As previously stated, the methodological approach is the BOW model, where the aim is to turn each word into a feature.

The process of transforming a sequence of text into individual features is also called *tokenization*, where each feature will be a token. This separation could be simply performed by separating words when there is white spaces in between. In this case a regular expression was used. Each character carries different information and because of that, for instance, ’word’ would be different of ’word.’, or even ’Word’.

No punctuation was considered, since in most of the cases it only adds noise to feature discrimination, and, for legislation categorization, it is believed that it does not introduce relevant information. However there are situations where some of the information would be lost or distorted, for example acronyms separated by dots such as ’E.U.’ will not be considered, or what are called *compound words* in Portuguese such as ’decreto-lei’ will end up as two separated words: ’decreto’ and ’lei’. Also, in Portuguese, pronouns can be concatenated to verbs with a separating hyphen, such as ’reservando-se’.

The hyphen verb/pronoun concatenation is much more frequent than the compound words, where in the first situation, no discriminative meaning is lost, since pronouns are not very interesting from the informative point of view. Still, if compound words lose their concept meaning, it does not imply that the words are lost, just mapped into separated features, with the inconvenient of the possibility of having different meanings. Other regular expression was tested which considered the words with hyphen, but no significant differences in classification results were obtained since the hyphen words are quite few compared to the rest of the feature space. Furthermore, since the used regular expression resulted in an more complex formula, the computation time for the preprocessing phase increased slightly.

All the words are converted to lower case so they all map into the same feature. Features of length of one are also discarded. This features are often resultant from abbreviations, acronyms or paragraph enumeration, which are common in law articles. No numeric characters were considered as features. Stop-word removal is performed as well, removing the most frequent words after the *tokenization* process, and before Feature Selection.

5.3 Document Term Matrix and Weighting

After text preprocessing, the documents, as explained in section 2.3, are represented by word vectors $d = \{t_1, ..., t_n\}$ where each word has a frequency. The word vector of each document is then resized to match the dimension $|T|$ of the whole vocabulary in the corpus. The words which do not appear in original vector, are just filled with zero. The document vectors are then joined and represented in a Document Term Matrix with dimension $|D| \times |T|$. All the computation such as term weighting, dimensionality reduction and classification methods is applied to this matrix. When selecting which documents are used for training or testing, the respective indices are selected and subsets are sent for computation. An illustrative example of the Document Term Matrix can be seen in table 5.4.

Table 5.4: Example of a Document Term Matrix.

	$Term_1$	$Term_2$	$Term_3$	$Term_4$
$Document_1$	1	0	3	2
$Document_2$	7	5	3	1
$Document_3$	4	2	0	9
$Document_4$	5	1	5	0

The TFIDF is used as weighting in all computations, as suggested in the literature. To support this choice and evaluate the importance of weighting in classification an experiment with the different components of TFIDF was performed in section 6.2.1.

5.4 Dimensionality Reduction, Classifiers and Evaluation

The testing framework was implemented with *Python* using popular science analysis libraries such as *Scikit-learn* and *numpy*.

The Supervised Classifiers implementations provided by the *Scikit-learn* framework which are used in this work are: *Multinomial Naive Bayes*, Support Vector Machines, Decision Trees, and K-Nearest Neighbours. The categories of the corpus, which are going to be classified, are not binary and consequently, MNB was chosen instead of the Multi-variate Bernoulli model. The K-NN was selected due to its simplicity and DT since it provides a different approach by hierarchically divide the feature space, which sometimes is referred as a good option for skewed classes. SVM is also included in order to validate the premise of the text data being linearly separable. Moreover, linear classifiers show the best classification performances in the literature.

Internal parameters such as the penalty parameter C for the SVM, the additive smoothing parameter for the MNB and the number of neighbours in K-NN are tuned for the best results, which are the ones presented here. DT classifier is computed with the Gini impurity criterion.

A k-fold cross validation methodology was used to evaluate the results of the different supervised experiments. The amount of labelled documents in the DRE corpus in some categories is reduced and for a cross folding validation is necessary at least *k* labelled documents in a category. Since the categories are quite unbalanced, the *stratified* scheme was chosen, where each splits maintain the proportions of the original classes. As explained in chapter 4, the corpus set is divided *k* times, and each partition is used as training and testing. A 10-fold cross validation was performed since it is the recommended choice for ensuring robustness of the results. Since the partitions are made in a random fashion and in some categories the available data is reduced, to ensure fairness to each classifier when comparing each other, each of the 10-fold partitions are the same used for training and testing other classifiers. Moreover, the 10-folds tests were repeated 3 times, where the results of the 10-folds are averaged, and then averaged again in respect to the repetitions.

For classification performance evaluation both F1 macro-averaged and micro-averaged are presented. This was motivated by the big unbalance in the number of documents in each class in the *DRE* corpus. Since macro-averaging weights each class score in an equal manner, if the biggest class has high score results and the smaller class performs with lower scores, than the macro-averaged score will be low, while the great majority of the documents was correctly classified. In the other way around, in similar situation, micro-averaging will yield a much higher score than macro-averaging, reflecting that the great majority of the documents was indeed correctly classified. Because of that, the macro-averaged score will indicate how the classification performed in respect to the classes and the micro-averaged score how the system performed globally in respect to the whole document set.

For an unsupervised clustering evaluation only the K-means was experimented, since in the literature no clustering method was found to produce significantly good results, and therefore only a superficial analysis of the problem is explored. For evaluation of the clustering performance the silhouette, completeness, homogeneity and V-measure were used.

Chapter 6

Experimental Results

6.1 Unsupervised Classification Experiment

Here it is presented the experiment performed with a clustering algorithm on the corpora as unlabelled data, but for which the true labels are known.

A clustering algorithm was applied to the corpora to verify how the categories perform under unsupervised circumstances. In first instance, the main purpose is to evaluate, in a simple form, the possibility of using a completely unsupervised algorithm to categorize the corpus. Moreover it is a form of analysing how documents in the categories are similar to each others.

6.1.1 Clustering with Fixed Number of Clusters

A fixed number of clusters, equal to the number of categories respective to each corpus, was used in the process of clustering with K-means algorithm. The idea is to observe if the clusters resemble to the original categories.

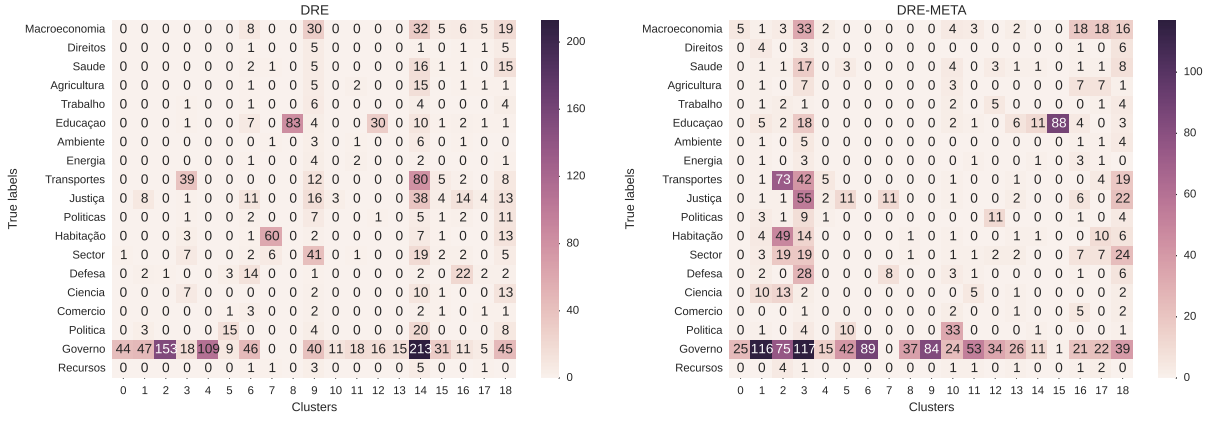
In table 6.1 the results for the multiple scores discussed in section 4.1 are presented. Some preliminary conclusions can be drawn from here. First the silhouette score is close to zero, meaning that the average distance between the samples within the clusters are close to the distances of the respective nearest clusters, which is a strong indication that the clusters overlap. The low homogeneity score suggests that the formed clusters contain multiple different categories and the low completeness score that the categories are not concentrated on the same clusters, thus leading to a low V-measure as well.

These conclusions can be seen in figure 6.1. For both corpora with the same categories, DRE and DRE-META, there is only one clear homogeneous cluster associated to only one category: ”Educação”, plus in the DRE corpus ”Habitação” also forms a clear homogeneous singular cluster. On the DRE-GAP corpus ”Condecorações” category shows an homogeneous cluster. However, none of the categories in any of the datasets produces complete clusters with mostly documents of the same class. Moreover, it is clear that the most populated classes often dominate the clustering process, being distributed by mostly all the clusters. The great diversity in such populated categories and the low discernibility resultant by vector distance comparisons of the K-means algorithm, probably due to the similarity be-

tween the contents of the documents, do not suggest clustering as a good option for unsupervised text classification.

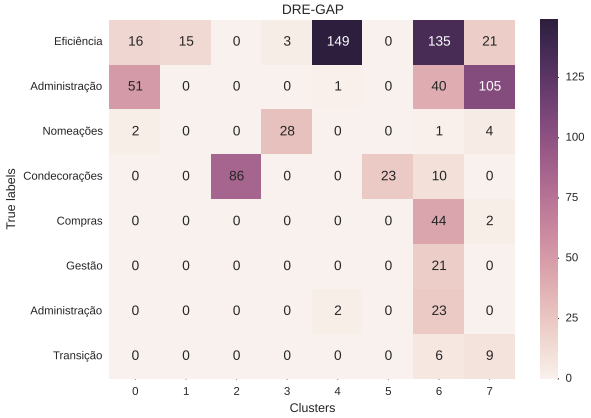
Table 6.1: Clustering with K-means.

Corpus	Sillouette	Homogeneity	Completeness	V-Measure
DRE	0.031	0.319	0.283	0.288
DRE-META	0.041	0.294	0.255	0.259
DRE-GAP	0.026	0.432	0.450	0.406



(a) DRE

(b) DRE-META



(c) DRE-GAP

Figure 6.1: Confusion Matrix plot for the K-means clustering.

6.2 Supervised Classification Experiments

In this section several experiments are performed with the supervised classifiers on the corpus with labelled data. Initially, weighting schemes are compared to understand which enhances the best scores of our datasets. Moreover, stemming technique is applied in order to improve the feature characteristics and reduce dimensionality. The impact of the reduction of the training size in classification scores is also demonstrated. Finally, techniques for feature transformation are also researched, such as feature selection with multiple metrics, and feature transformation with LSI.

6.2.1 Baseline and Weighting Impact Evaluation

Each component of the TFIDF weighting scheme was tested with different classifiers. Term-Frequency (TF), TFIDF without the log and with, which is denoted by $TFIDF_{log}$, are evaluated in this experiment. To each different weight scheme, the Euclidean norm is applied and evaluated as well.

As can be seen in table 6.2, each component of the TFIDF weighting scheme, described in the subsection 2.3.2, contributes in many cases for an increase of the classification performance.

Table 6.2: Impact of Feature Weighting in the corpora.

		SVM		MNB		KNN		DT	
Corpus	Weighting	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$
DRE	TF	0.800	0.553	0.777	0.543	0.675	0.424	0.595	0.394
	Norm(TF)	0.838	0.649	0.820	0.578	0.759	0.497	0.613	0.430
	TFIDF	0.830	0.597	0.784	0.576	0.685	0.458	0.601	0.399
	Norm(TFIDF)	0.853	**0.660**	**0.823**	**0.609**	0.791	0.541	**0.621**	**0.436**
	$TFIDF_{log}$	0.851	0.606	0.787	0.574	0.647	0.421	0.607	0.407
	Norm($TFIDF_{log}$)	**0.860**	0.647	0.806	0.566	**0.799**	**0.548**	0.613	0.419
DRE-META	TF	0.825	0.611	0.778	0.573	0.697	0.420	0.703	0.517
	Norm(TF)	0.813	0.620	**0.809**	0.584	0.754	0.491	0.662	0.462
	TFIDF	0.836	0.636	0.772	0.577	0.701	0.455	0.694	0.506
	Norm(TFIDF)	0.818	0.634	0.807	**0.605**	0.782	0.534	0.673	0.477
	$TFIDF_{log}$	**0.839**	**0.638**	0.768	0.570	0.696	0.454	**0.707**	**0.522**
	Norm($TFIDF_{log}$)	0.830	0.632	0.808	0.600	**0.788**	**0.534**	0.675	0.483
DRE-GAP	TF	0.858	0.758	0.863	0.775	0.776	0.655	0.746	0.647
	Norm(TF)	0.875	0.791	**0.881**	0.774	0.853	**0.745**	**0.769**	**0.683**
	TFIDF	0.862	0.761	0.861	0.754	0.697	0.539	0.763	0.660
	Norm(TFIDF)	**0.890**	**0.809**	0.880	0.802	0.861	0.739	0.766	0.680
	$TFIDF_{log}$	0.875	0.796	0.871	**0.808**	0.640	0.501	0.752	0.644
	Norm($TFIDF_{log}$)	0.887	0.809	0.878	0.782	**0.862**	0.721	0.760	0.671

Clearly the best classifier is the SVM, as seen in most of the literature comparisons, where it performs the best for both $F1_m$ and $F1_M$ scores, outperforming the other classifiers for both datasets. NB is the second which provides best results, followed by KNN and lastly DT.

The most striking results go for the *DRE-META* corpus, where highest scores yield performances close to the *DRE* corpus, even with a feature set much more smaller than its full counterpart. Furthermore by observing the $F1_M$ score, it can be seen that for SVM classifier, it actually has better performances, with non normalized weights, in the DRE-META than in the DRE corpus. It is interesting to note that the DRE-META corpus, even with a much smaller feature space, being composed only by the subjects and small summary of the documents, allow to classify with similar results than it full

document counterpart dataset, the DRE corpus. Being the $F1_M$ score higher than the $F1_m$, means that the categories, globally, are better classified in the DRE-META than in the DRE corpus. This could indicate that the former contains features which are more informative for the less populated classes than the in the later. This can be observed in figure 6.2, where some of the categories which had the worse performances in DRE, shown slightly better results in DRE-META. The DRE-GAP corpus presents high classification scores, however it is due to its class distribution, which has less classes to discern between, and also highly populated classes. The categories which have more document examples for training show better results. This can be easily observed in figure 6.2, where the categories which perform the worst are associated to the ones that have very few documents.

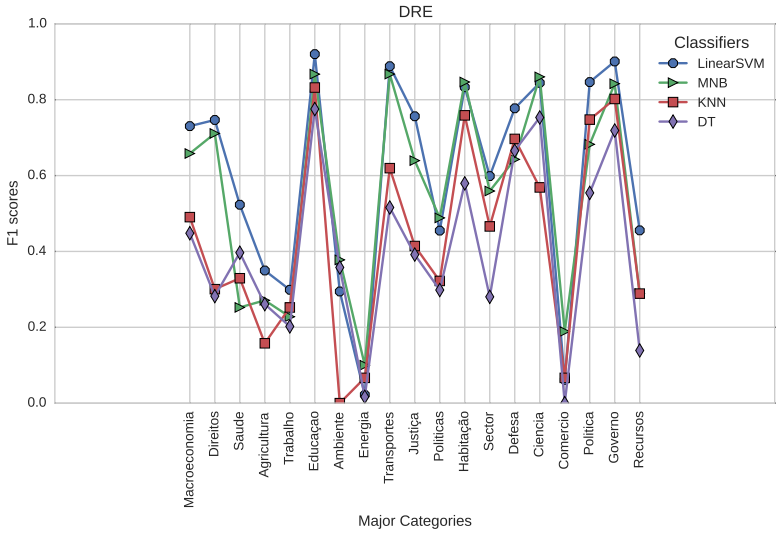
In general, for the DRE corpus, the Euclidean norm as weighting scheme improves its results, however in the DRE-META corpus it can be seen that the norm only worsens the results, with exception of the KNN classifier. This might be due to the very reduced amount of features, not only in diversity but in terms of counts. Since the term frequencies are much lower, and consequently close to each other, the discernibility is probably lost in the multiplication of very small values. The same logic applies to why the TF in some cases performs better than the $TFIDF$ counterparts.

The best weighting schemes, as expected, are the $TFIDF$ and the $TFIDF_{log}$. Even though the highest score obtained was the $F1_m$ for the normalized $TFIDF_{log}$, the highest score for the $F1_M$ was with the normalized $TFIDF$, both with SVM. Here lies the problem of which score to maximize. Since the $TFIDF_{log}$ actually harms the performance in some classifiers for both micro and macro averaged scores, in comparison with the $TFIDF$ without log, it is considered that the weighting scheme which performs the best is indeed the $TFIDF$, therefore, it is the scheme that is applied to the Document Term Frequency matrices in the following experiments.

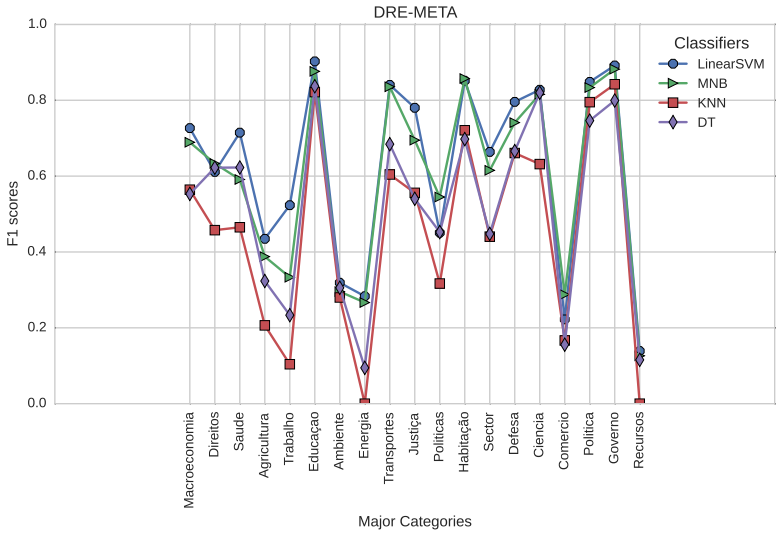
Computation Times

The execution times for each classifier can be observed in figure 6.3. The fastest classifier, both in training and testing is the MNB, which can be attributed to its simplicity. SVM has the second highest training time but with test times similar to the MNB. The KNN classifier benefits in training time since, as presented in chapter 3.2, it only needs to store the feature vectors. However in training, it is the worst performer of all classifiers. Smaller datasets result in lower computation times since the quantity of data which the algorithm deals with is also lower. The DT classifier is the worst in terms of training execution time due to the heavy cost of constructing and storing the necessary classification tree.

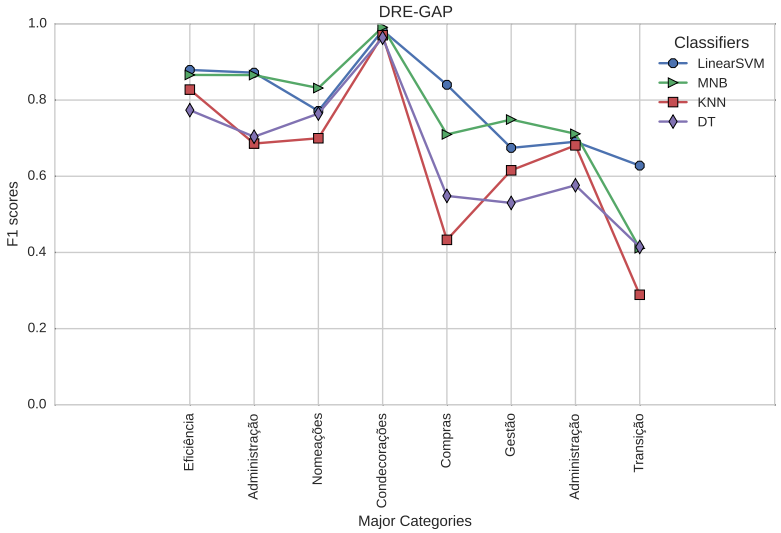
If case of a deployment where time execution performance matters, MNB is an interesting choice, with low training and test times. However for TC in the legal domain, the publishing frequency and quantity does not require hard execution times. Being the classification performance of the SVM higher than all the other classifiers, and its execution time close to the two other best performers (in classification and times) makes it the best choice.



(a) DRE

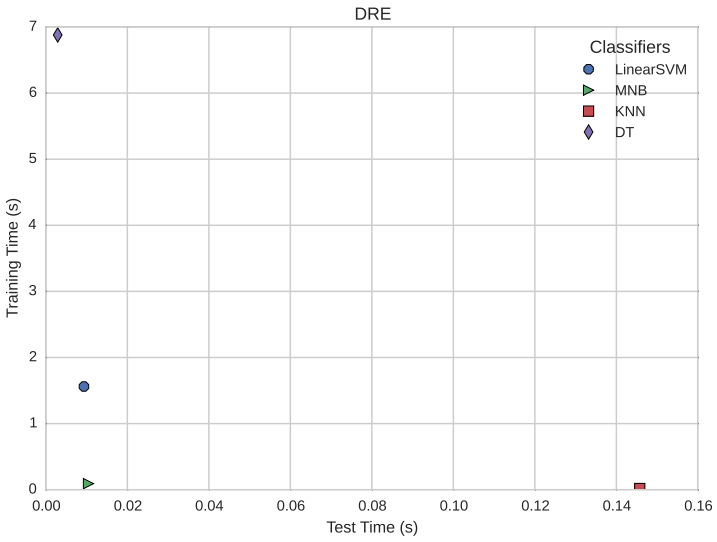


(b) DRE-META

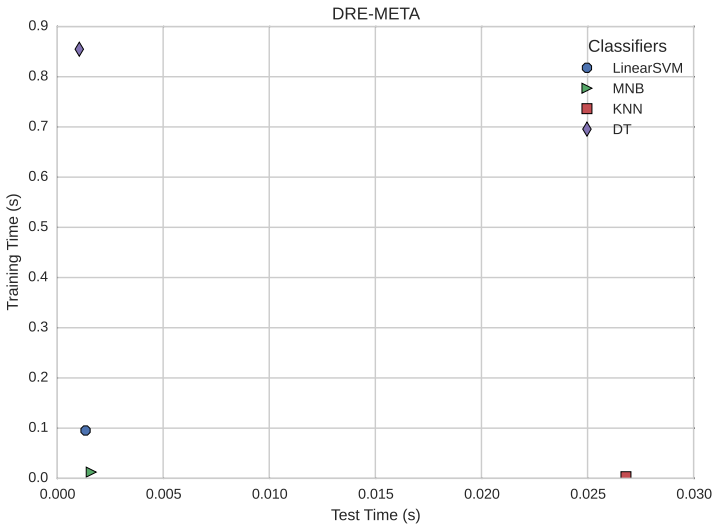


(c) DRE-GAP

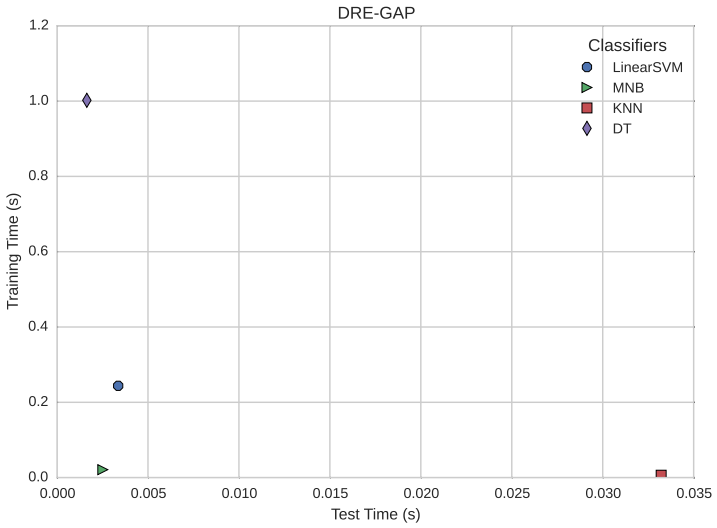
Figure 6.2: F1 scores for each category of each corpus and classifier with features weighted with $||TFIDF||$.



(a) DRE



(b) DRE-META



(c) DRE-GAP

Figure 6.3: Computation time for training and testing for multiple classifiers with features weighted with $||TFIDF||$ for each corpus.

6.2.2 Stemming Evaluation

In this experiment, feature transformation with stemming using suffix removal was applied to the datasets. The experiments were performed with the normalized TFIDF. To facilitate comparison the baseline results are repeated and summarized in table 6.3.

Table 6.3: Impact of Feature Stemming in the corpora.

	LinearSVM		MNB		KNN		DT	
Corpus	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$
DRE	0.853	0.660	0.823	0.609	0.791	0.541	0.621	0.436
DRE Stemmed	0.821	0.585	0.770	0.540	0.705	0.437	0.600	0.400
DRE-META	0.818	0.634	0.807	0.605	0.782	0.534	0.673	0.477
DRE-META Stemmed	0.814	0.617	0.786	0.558	0.715	0.468	0.700	0.516
DRE-GAP	0.890	0.809	0.880	0.802	0.861	0.739	0.766	0.680
DRE-GAP Stemmed	0.868	0.784	0.856	0.760	0.798	0.672	0.731	0.655

There is no positive impact in the stemmed versions of corpus with big feature spaces, such as the DRE and DRE-GAP, meaning that in terms of classification performance stemming does not seem to bring any clear advantages to high dimensional feature sets. However, in the DRE-META corpus with DT, there are some improvements, nonetheless, with small expression. This could be explained by the principal purpose of the stemming, which is mapping different words into their root feature (word), benefiting by having more *singular* features represented in different documents, instead of multiple features with the same meanings. The SVM presents again the best performance, followed by the MNB, KNN and finally DT.

In terms of feature reduction, in the DRE corpus, it resulted in a feature space of 16694 unique features, which represents a reduction of almost 55%. In the case of DRE-META, the stemming process resulted in a feature space of 3064 unique features, a reduction of 37% of the dimensionality. For the DRE-GAP there is a reduction of 57% with 8219 unique features.These reductions in the feature space are quite significant with only marginal impact in the performance of the classification.

Computation Time

Since the stemming algorithm is applied to every word in the corpus, the bigger the number of words in it, the bigger will be the processing time. In figure 6.4 can be found the preprocessing times of the corpus. As expected, due to its higher number of features, the DRE corpus had the greater increase in processing time, almost 6 times its original execution.

Depending on the deployment necessities, the gain in the classifier execution times might not justify the increase in the preprocessing time for large datasets. However, for smaller corpus which have a lower feature space, at the cost of a slight increase of computation time, small gains could be achieved.

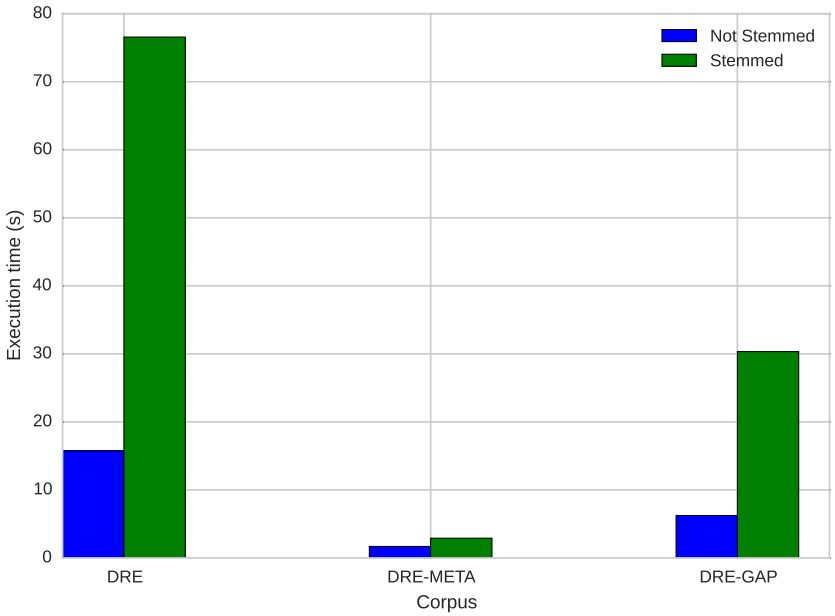


Figure 6.4: Corpus processing times with and without stemming.

6.2.3 Impact of the Training Size in Classification Performances

To better understand the effect of reduced datasets in classification performance, multiple trials with different sizes for the training and testing were produced. In this experiment, instead of a 10-fold cross-validation approach, a random sub-sampling was used with the stratified variant, where the proportions of the classes are maintained. Since the sizes for the training and testing sets have to be fixed beforehand, a percentage of the size was fixed and multiple trials executed. The percentage presented in table 6.4 is respective to the training size and the total remainder is always used for testing.

As expected, the classification performances increase along with the training size of the set, with some particular exceptions. The more information the classifier receives during training, the more apt it will be when dealing with unobserved data. Naturally, it is not an unbounded growth, being expected to smooth at the point where it reaches the classifier limitations and the data itself.

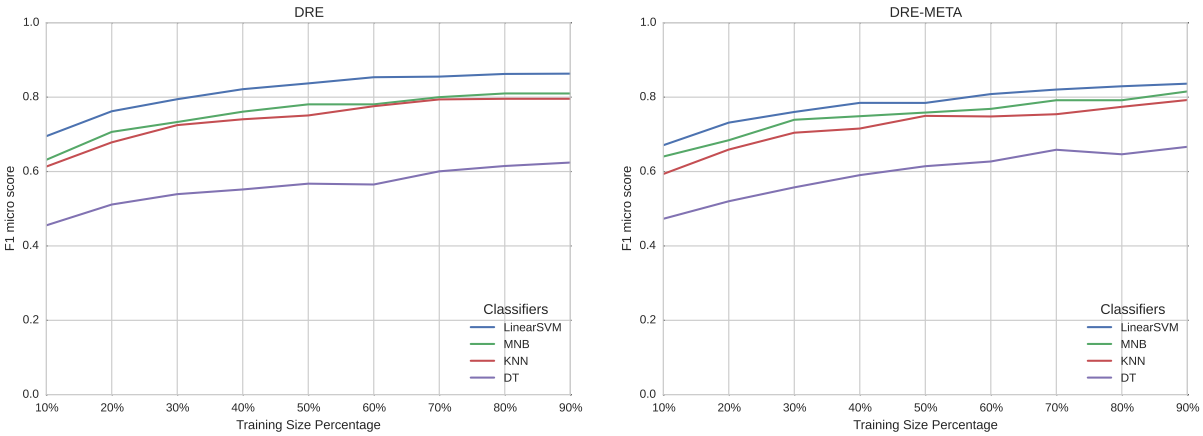
For the DRE corpus it can be seen that it starts to reach its maximum values at around 80% of the training size for the SVM, MNB and KNN. While on DRE-META corpus only SVM seems to reach the maximum possible score. This does not imply that these are the maximum scores for the classifiers in this corpus, only that this might be the best possible performances achieved with the available labelled data for training.

There are some particular peaks in the classification performances. For instance, the DRE-GAP corpus with a train size of only 50% reaches the highest $F1_m$ score. This could be due to the selection of "lucky" subsets which contains the most informative features. Still, since multiple random trials were run, it is more probable that a smaller training size avoids over-fitting the classifier and, therefore, allowing to better generalize for unseen documents.

The classification score F_m is less affected by the training size, while the F_M increase more significantly. Since the F_m score expresses, in certain way, the amount of corrected classified documents in a global scope, and the biggest percentage of those classifications are governed by the most populated categories, the scores tend to skew and reflect those categories. For instance, 10% of the training size for the DRE corpus corresponds to 83 documents in the most populated category. This means that with only 10% of training size it already has more documents than the total documents in the remaining categories in the same corpus. This can be easily observed in figure 6.5, which shows the importance of having well populated categories to allow the classifiers to perform satisfactorily.

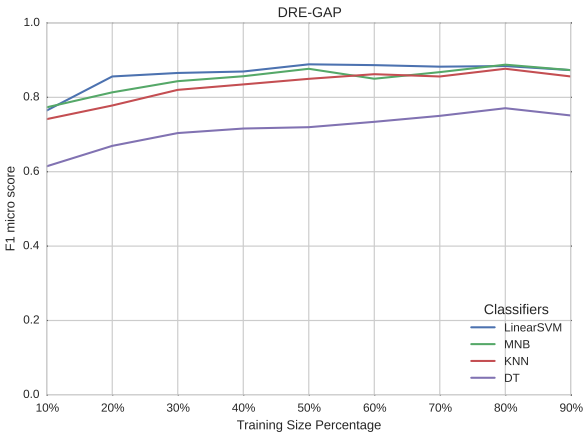
Table 6.4: Impact of the multiple train size in classification performances for different corpora and classifiers.

			Training Sizes								
Corpus	Classifiers	avg	10%	20%	30%	40%	50%	60%	70%	80%	90%
DRE	LinearSVM	F_M	0.373	0.502	0.520	0.600	0.604	0.630	0.632	**0.663**	0.646
		F_m	0.695	0.762	0.794	0.821	0.837	0.854	0.855	0.862	**0.863**
	MNB	F_M	0.290	0.423	0.461	0.514	0.522	0.523	0.542	0.577	0.607
		F_m	0.631	0.707	0.733	0.761	0.781	0.781	0.800	0.810	0.810
	KNN	F_M	0.293	0.390	0.445	0.477	0.486	0.505	0.549	0.533	0.535
		F_m	0.613	0.678	0.725	0.741	0.751	0.776	0.794	0.796	0.796
	DT	F_M	0.205	0.307	0.329	0.348	0.370	0.362	0.393	0.416	0.418
		F_m	0.455	0.511	0.539	0.552	0.567	0.565	0.601	0.615	0.624
DRE-META	LinearSVM	F_M	0.386	0.481	0.503	0.579	0.568	0.574	0.637	0.615	**0.639**
		F_m	0.671	0.732	0.760	0.785	0.785	0.808	0.820	0.829	**0.836**
	MNB	F_M	0.360	0.438	0.506	0.525	0.507	0.558	0.552	0.570	0.617
		F_m	0.640	0.684	0.739	0.749	0.759	0.769	0.792	0.792	0.815
	KNN	F_M	0.257	0.350	0.410	0.441	0.479	0.490	0.497	0.514	0.546
		F_m	0.593	0.659	0.704	0.716	0.750	0.748	0.754	0.774	0.792
	DT	F_M	0.230	0.298	0.313	0.398	0.422	0.410	0.446	0.436	0.488
		F_m	0.473	0.520	0.557	0.590	0.614	0.627	0.659	0.646	0.666
DRE-GAP	LinearSVM	F_M	0.524	0.711	0.753	0.774	**0.817**	0.809	0.803	0.817	0.798
		F_m	0.765	0.856	0.866	0.869	**0.889**	0.887	0.882	0.884	0.873
	MNB	F_M	0.616	0.700	0.751	0.762	0.778	0.740	0.789	0.797	0.791
		F_m	0.773	0.813	0.843	0.857	0.877	0.850	0.868	0.888	0.873
	KNN	F_M	0.480	0.626	0.715	0.725	0.735	0.740	0.739	0.757	0.746
		F_m	0.741	0.778	0.820	0.835	0.850	0.862	0.856	0.877	0.856
	DT	F_M	0.421	0.559	0.614	0.617	0.640	0.646	0.673	0.702	0.695
		F_m	0.615	0.670	0.704	0.716	0.720	0.734	0.750	0.771	0.751



(a) DRE

(b) DRE-META



(c) DRE-GAP

Figure 6.5: F1-micro scores for different corpus and classifiers with multiple training sizes.

6.2.4 Evaluation of Feature Selection Methods

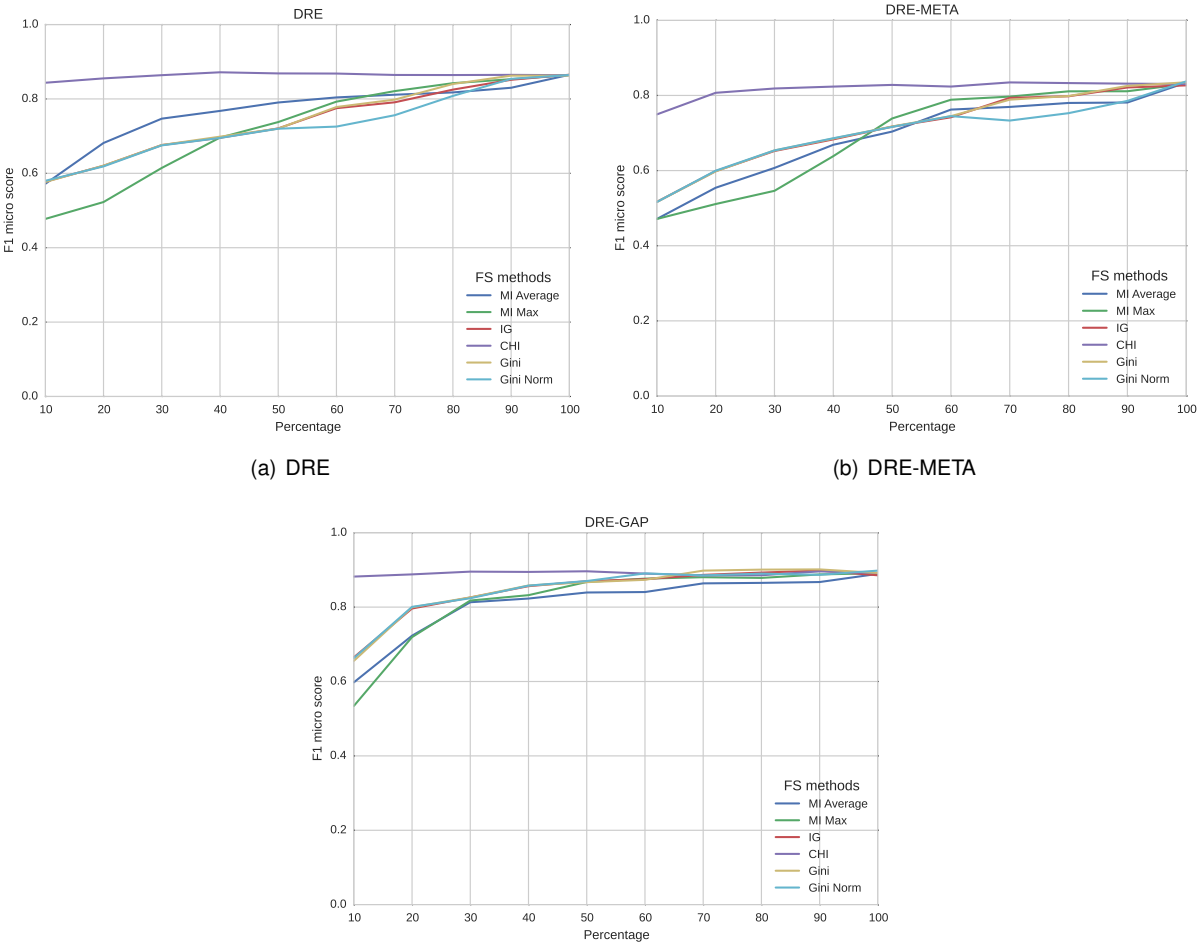
Since selecting features may improve results and avoid over-fitting an experiment with the methods presented in section 2.4.1 was performed, with exception of the χ^2 where only the averaged version results are presented. Concerning the classifiers, feature selection metrics were performed only with SVM for several reasons. The computation times for all the classifiers with multiple feature selection metrics took considerable time, plus it would require considerable space to present the results here. Furthermore, SVM so far shows to be the best classifier and the goal is to maximize the classification scores. The results can be found in table 6.5 and the classification performances for each feature selection metric in figure 6.7.

All the maximum scores achieved with χ^2, with exception of DRE-META and DRE-GAP for the F_m scores, were higher than the baseline scores. Still, overall none of the methods provided significantly better results than the baseline classifications. However, χ^2 feature selection method showed to be a powerful method, allowing performances similar to the full set of features. Only 20% of the most significant feature set for the DRE, DRE-META and DRE-GAP was necessary to achieve similar classification scores of the original set.

			Feature Percentage									
Corpus	Methods	avg	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
DRE	MI Average	F_M	0.375	0.486	0.554	0.577	0.573	0.615	0.596	0.601	0.616	0.657
		F_m	0.571	0.685	0.748	0.765	0.787	0.803	0.811	0.818	0.834	0.865
	MI Max	F_M	0.236	0.372	0.467	0.521	0.558	0.593	0.633	0.640	0.632	0.658
		F_m	0.476	0.523	0.619	0.695	0.738	0.794	0.823	0.845	0.851	0.865
	IG	F_M	0.260	0.334	0.415	0.459	0.484	0.546	0.568	0.628	0.660	0.661
		F_m	0.578	0.621	0.676	0.698	0.720	0.775	0.792	0.829	0.850	0.865
	CHI	F_M	0.673	**0.677**	0.669	0.660	0.660	0.657	0.661	0.657	0.651	0.661
		F_m	0.845	0.855	0.863	**0.870**	0.866	0.867	0.867	0.865	0.864	0.867
	Gini	F_M	0.261	0.335	0.412	0.456	0.485	0.544	0.576	0.633	0.668	0.659
		F_m	0.578	0.620	0.677	0.698	0.721	0.781	0.798	0.837	0.863	0.865
	Gini Norm	F_M	0.257	0.338	0.409	0.457	0.486	0.589	0.591	0.615	0.683	0.660
		F_m	0.578	0.622	0.675	0.696	0.721	0.729	0.762	0.807	0.852	0.865
DRE-META	MI Average	F_M	0.313	0.399	0.437	0.503	0.527	0.553	0.584	0.594	0.594	0.628
		F_m	0.477	0.555	0.607	0.674	0.702	0.761	0.768	0.780	0.780	0.832
	MI Max	F_M	0.193	0.333	0.398	0.491	0.550	0.598	0.596	0.609	0.619	0.626
		F_m	0.472	0.512	0.545	0.635	0.736	0.790	0.798	0.812	0.811	0.828
	IG	F_M	0.141	0.276	0.351	0.429	0.502	0.528	0.587	0.591	0.636	0.631
		F_m	0.516	0.599	0.652	0.686	0.718	0.743	0.794	0.798	0.823	0.829
	CHI	F_M	0.601	**0.646**	0.625	0.634	0.630	0.626	0.625	0.634	0.629	0.628
		F_m	0.749	0.809	0.820	0.824	0.826	0.828	0.830	**0.836**	0.831	0.828
	Gini	F_M	0.142	0.275	0.349	0.425	0.501	0.528	0.573	0.591	0.647	0.629
		F_m	0.516	0.599	0.652	0.685	0.717	0.744	0.789	0.797	0.824	0.833
	Gini Norm	F_M	0.141	0.278	0.352	0.426	0.500	0.526	0.567	0.580	0.620	0.627
		F_m	0.516	0.599	0.653	0.686	0.717	0.743	0.737	0.756	0.786	0.837
DRE-GAP	MI Average	F_M	0.527	0.635	0.710	0.737	0.745	0.749	0.766	0.772	0.798	0.837
		F_m	0.601	0.724	0.814	0.824	0.840	0.846	0.856	0.865	0.872	0.892
	MI Max	F_M	0.541	0.671	0.769	0.765	0.779	0.790	0.808	0.793	0.823	0.839
		F_m	0.535	0.711	0.819	0.833	0.871	0.874	0.885	0.881	0.888	0.892
	IG	F_M	0.521	0.597	0.631	0.676	0.706	0.765	0.802	0.810	0.813	0.834
		F_m	0.653	0.796	0.824	0.857	0.869	0.873	0.890	0.894	0.898	0.887
	CHI	F_M	**0.845**	0.836	0.832	0.832	**0.842**	0.834	0.832	0.829	0.841	0.816
		F_m	0.886	0.887	**0.898**	0.891	0.898	0.893	0.886	0.885	0.896	0.890
	Gini	F_M	0.523	0.605	0.631	0.679	0.704	0.769	0.823	0.830	0.825	0.838
		F_m	0.660	0.799	0.825	0.859	0.868	0.874	0.898	0.898	**0.901**	0.893
	Gini Norm	F_M	0.524	0.597	0.633	0.673	0.702	0.812	0.839	0.837	0.837	0.823
		F_m	0.657	0.799	0.825	0.856	0.869	0.891	0.887	0.892	0.888	0.894

Table 6.5: Multiple feature selection methods for the SVM classifier with different corpus.

Figure 6.6: F1 micro scores for each feature selection method of each corpus with SVM.



(a) DRE

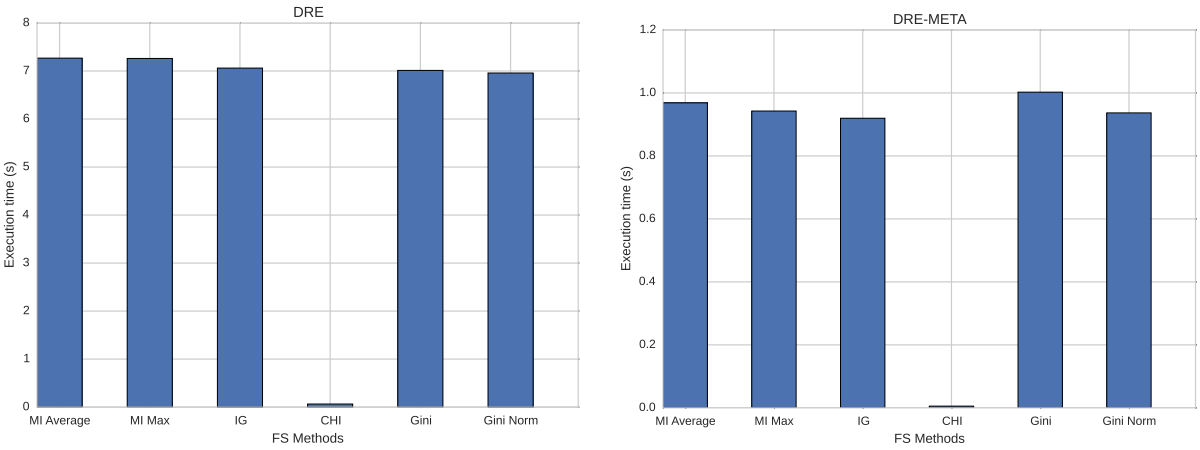
(b) DRE-META

(c) DRE-GAP

Figure 6.7: F1 micro scores for each feature selection method in each corpus using SVM.

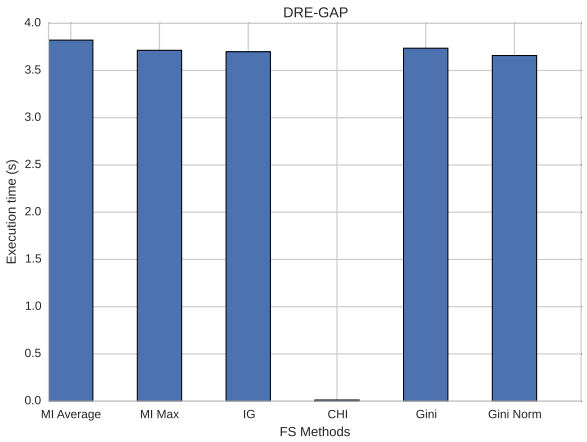
Computation Time

In terms of processing the feature selection methods share similar computation times, with exception of the χ^2 which takes considerable less time. The results can be observed in figure 6.8. It is not really fair to compare the performances since all the other metrics besides χ^2 share a common implementation, while the later uses an optimized one.



(a) DRE

(b) DRE-META



(c) DRE-GAP

Figure 6.8: Computation time of the feature metrics in each corpus.

6.2.5 Feature Transformation with LSI

More than just a dimensional reduction method, LSI aims to capture the underlying latent semantics in the texts. To evaluate its effect in the classification, multiple sizes, or in other words, singular vectors, were selected in this experiment. In the literature usually 200 to 500 principal vectors are selected. Since the purpose is to capture the most information with the lowest amount of dimensions, a set of 100 to 2000 dimensions were used, as can be seen in table 6.6. Since MNB does not support negative features, which the SVD algorithm produces when computing the principal vectors, it was not included in this experiment.

It is interesting to note that with a very reduced feature set, the classifiers are able to achieve performances close to the original non-transformed features. In overall, F_m score has no significant improvements, while on corpus with fewer features, such as the DRE-META, shown to be marginally worse. It is possible to conclude that for corpus with low dimensional spaces, LSI does not provide great advantages since the classification performance requires bigger dimensions to capture information similar to the original. However, for datasets as the DRE and DRE-GAP, the LSI with only 2000 dimensions, a feature dimensional reduction close to 95% for the DRE and 90% for the DRE-GAP, provides similar results as the original dataset.

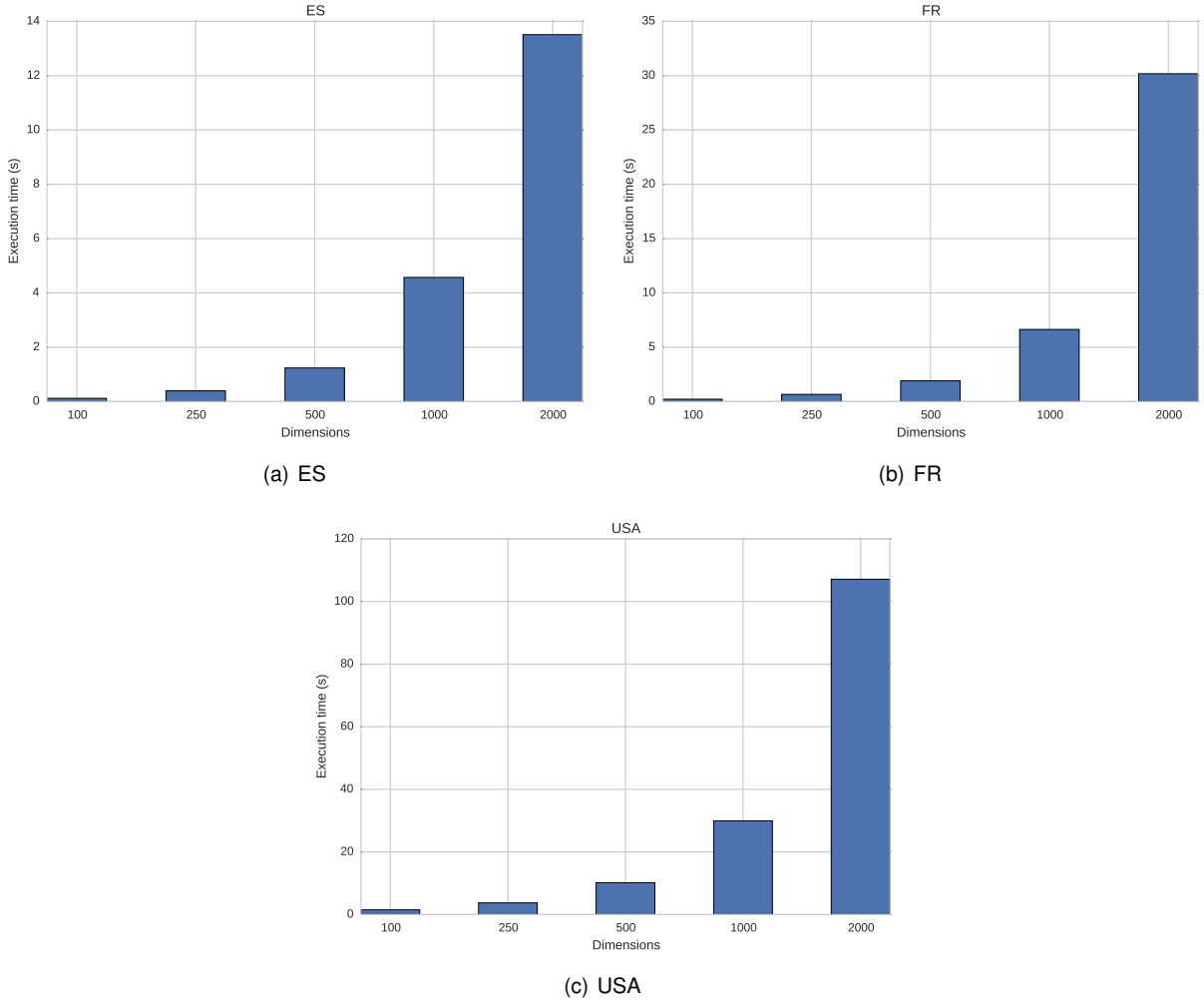
Table 6.6: Feature transformation with LSI with pre-selected dimensions for multiple corpus and classifiers.

			Dimensions				
Corpus	Classifiers	avg	100	250	500	1000	2000
DRE	LinearSVM	F_M	0.612	0.653	**0.668**	0.667	0.664
		F_m	0.813	0.838	0.845	0.860	**0.867**
	KNN	F_M	0.532	0.545	0.571	0.538	0.561
		F_m	0.790	0.804	0.819	0.795	0.802
	DT	F_M	0.395	0.367	0.360	0.325	0.305
		F_m	0.611	0.583	0.564	0.548	0.517
DRE-META	LinearSVM	F_M	0.537	0.577	0.616	**0.630**	0.628
		F_m	0.740	0.775	0.790	0.816	**0.831**
	KNN	F_M	0.498	0.525	0.534	0.529	0.533
		F_m	0.730	0.764	0.787	0.780	0.775
	DT	F_M	0.348	0.319	0.323	0.283	0.280
		F_m	0.539	0.529	0.530	0.499	0.495
DRE-GAP	LinearSVM	F_M	0.812	0.828	0.830	0.805	**0.839**
		F_m	0.879	0.889	0.882	0.885	**0.896**
	KNN	F_M	0.803	0.769	0.719	0.734	0.734
		F_m	0.880	0.874	0.860	0.866	0.867
	DT	F_M	0.694	0.694	0.677	0.643	0.654
		F_m	0.777	0.782	0.765	0.746	0.746

Computation Time

In figure 6.9 can be seen the processing times necessary for each dimension and each corpus. It can be observed that computation time increases proportionally to two factors: the full dimension of the corpus,

and the dimension chosen for the data. The corpus size influences the computational time because there is more data to compute all the singular vectors, before sorting and truncating them. The latter factor is due to the time required to create new data in memory which grows proportionally with the dimensions chosen.



(a) ES

(b) FR

(c) USA

Figure 6.9: Computation time for the LSI algorithm on DRE, DRE-META and DRE-GAP corpora.

Chapter 7

Conclusions

Classification of legislation texts is a task which requires a great amount of time and specialized labour. In this work, ML methods were explored in order to seek the possibility of doing the classification work automatically, or at least to facilitate this tasks. TC is a field of study which has gained a lot of attention in the late years due to the growth of Internet and data collection, and consequently also the necessity of classifying and organizing that information. Legislation is one of the many fields that can benefit from these techniques.

To evaluate the possibility of categorizing legislation with ML, we discussed multiple preprocessing methods and classifiers. Here some of those were tested and compared between each other.

As document representation, BOW methodology was applied and each word was treated as a feature. This feature representation showed to be able retain the corpus characteristics well enough to provide reasonable classification scores. Two paradigms were tested. Unsupervised clustering by exploring the implicit structure of the corpora using K-means and supervised classification using the classifications provided by experts was evaluated with SVM, K-NN, MNB and DT.

Three different corpus where constructed with the collected dataset from the legislation available in the DRE website. The most relevant, DRE corpus, contained the full text of the law documents. Two other datasets were created based on it: the DRE-META and DRE-GAP corpus. The DRE-META contained only a small summary of the publications and ministry that published it as features. The DRE-GAP was created using the most populated major category of DRE dataset. The DRE and DRE-META documents were classified using the same set of major category labels, while in the DRE-GAP corpus the documents were classified using the minor categories labels.

Common techniques of dimensionality reduction were also tested with the goal of increasing the classification performances, such as stemming, feature selection and feature transformation. Stemming algorithm is a common dimensionality reduction technique in BOW document representations, since it allows to merge multiple features into the same root word. Still, it did not show to provide improvements in the classification accuracy of the documents, but it was able to reduce considerably the feature set, while maintaining scores close to the baseline. LSI and feature selection with the χ^2 metric show marginally better results in some cases. However no significant impact was observed in terms of classi-

fication results. Both techniques shown a considerable power as dimensionality reduction tools allowing to achieve similar classification scores with less than 10% of the original feature set. It should be noted that LSI is an unsupervised algorithm for dimensionality reduction while χ^2 is a supervised method. χ^2, in contrary to LSI, allows easily to identify the words which are the most informative to a category, which could be helpful in the creation of an engine of topic suggestion words to a user.

The unsupervised clustering experiments in the DRE and DRE-META corpora did not show clusters that allowed to classify the multiple major categories of the documents. The experiments shown that using this unsupervised algorithm it was not possible to classify the legislation documents with reasonable performances. Supervised algorithms demonstrated, as expected, more success in correctly classifying the documents. The best classifier is, without doubt, the SVM for the supervised classification. It is interesting to note that SVM is a linear classifier, being clear that the text data is indeed sparse enough and therefore easily separated with only linear algorithms. SVM performance follows closely the tendency observed in the TC literature. MNB which makes a gross assumption of independence among features, often called in the literature "weak classifier", also shows good results being a classifier that is simple and easy to implement.

It was expected lower classification scores in the DRE-META corpus since it has less unique features compared to the DRE corpus. Even so, it was possible to achieve classifications scores in the DRE-META corpus which were close to the DRE corpus. It was demonstrated that it is not a matter of quantity of (unique) features, but instead, how much information they provide to the category, and consequently, to the training of the classifier. With the DRE-GAP corpus it was shown that, if provided adequate number of classified document samples for training, the classifiers are able to correctly classify documents which belong to categories within similar topics. This allows the possibility of a finer and detailed classification of the documents.

Furthermore, computational execution times were also compared. In the classifier comparisons, MNB showed to be a fast solution, while DT resulted in excessive training times and K-NN and high testing times as well. SVM demonstrated balanced computation times, relative to the other classifiers. Stemming and LSI increased considerably the execution times but provided significant feature set reductions, plus in some cases increasing classification performances.

It has to be noted, that the coded DRE corpus is significantly skewed. One of the biggest difficulties of this work was to provide reasonable classification scores at the same level of those found in the literature. Often TC is performed with much more documents examples for training the classifier. Other methodologies with ensemble, boosting and bagging methods were tested but due to lack of space and success in improving the results, those are not presented in this work. Still, it was possible to achieve high classification accuracies in some of the well populated classes. The classes for which there are fewer pre-classified examples, the results do not achieve desirable classification scores.

Overall, the classification of legislation has proved to be possible with satisfying results. Some of the categories did not reached the same levels of success, however, it became clear that it is fundamental to have a good sample of pre-classified examples to produce acceptable classification results.

7.1 Future Work

For this work a database was collected with a considerable sample of the Portuguese legislation. Methods such as the semi-supervised techniques allow to merge the best of the two worlds: using low labelled data and incorporating cheap unlabelled data. There are many of state of the art methods which could be used with all this data. However, these algorithms usually require complex implementations and high amounts of unlabelled data to experiment with. Such implementations have to take into consideration the high dimensionality of the data and its heavy computational memory requirements.

Techniques based on unsupervised methods do not show promising results, but incorporating other information such as meta data or the document structure, and weighting it accordingly as features could be another path to take into account. Also, topic modelling methods, even with its broad interpretations, could provide other relevant information for researchers. Unsupervised algorithms are yet a big area of study, and a conjunction of different approaches might lead to better results. Still, for categorization purposes such methods do not seem to provide competitive classification performances with respect to the supervised methods.

Other approaches could be explored using the intrinsic characteristics of the DRE dataset. Many documents make references to other documents, in a fashion similar to the hyperlinks in web-pages or citations from scientific papers. Information from the linked documents could be exploited, identical to the work of Yang et al. [80]. For instance, when a document links another, which might be due to being an extension to other law, correcting one or under the effect of multiple legislations, the features or part of them could be included in the original feature set of the document.

Bibliography

[1] J. Grimmer and B. M. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21:267–297, 2013.

[2] J. B. Slapin and S.-O. Proksch. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722, 2008.

[3] J. Grimmer. A bayesian hierarchical topic model for political texts: Measuring expressed agendas in senate press releases. *Political Analysis*, 18(1):1–35, 2010.

[4] M. Laver, K. Benoit, and T. College. Extracting policy positions from political texts using words as data. *American Political Science Review*, pages 311–331, 2003.

[5] B. C. Burden and J. N. R. Sanberg. Budget rhetoric in presidential campaigns from 1952 to 2000. *Political Behavior*, 25(2):97–118, 2003.

[6] B. M. Stewart and Y. M. Zhukov. Use of force and civil–military relations in russia: an automated content analysis. *Small Wars & Insurgencies*, 20:319–343, 2009.

[7] P. Francesconi. Automatic classification of provisions in legislative texts. *Artificial Intelligence and Law*, 15(1):1–17, 2007.

[8] S. Purpura and J. F. Kennedy. Automated classification of congressional legislation. In *In Proceedings of Digital Government Research (dg.o*, 2006.

[9] R. Feldman and J. Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[10] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180. Association for Computational Linguistics, 2003.

[11] A. Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142. Philadelphia, USA, 1996.

[12] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. Text classification using machine learning techniques. *WSEAS Transactions on Computers*, 4(8):966–974, 2005.

[13] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[14] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

[15] M. Lan and H. boon Low. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *In Posters Proc. 14th International World Wide Web Conference*, pages 1032–1033, 2005.

[16] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 37–50. ACM, 1992.

[17] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 148–155. ACM, 1998.

[18] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In *Advances in Information Retrieval*, pages 181–196. Springer, 2004.

[19] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. ACM, 1996.

[20] G. Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, 2003.

[21] F. Sebastiani. Machine learning in automated text categorization. *ACM COMPUTING SURVEYS*, 34:1–47, 2002.

[22] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

[23] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, Mar. 1986.

[24] Z. Zheng, X. Wu, and R. Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89, 2004.

[25] D. Mladenic and M. Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 258–267. Morgan Kaufmann Publishers Inc., 1999.

[26] M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 659–661. ACM, 2002.

[27] M. F. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A. G. Chin, editor, *Text Databases and Document Management*, pages 78–102. IGI Global, 2001.

[28] L. Galavotti, F. Sebastiani, and M. Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In *Research and Advanced Technology for Digital Libraries*, pages 59–68. Springer, 2000.

[29] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-454-5.

[30] V. M. Orengo and C. Huyck. A stemming algorithm for the portuguese language. In *String Processing and Information Retrieval, 2001. SPIRE 2001. Proceedings.Eighth International Symposium on*, pages 186–193, 2001.

[31] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[33] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 99th edition, 1975. ISBN 047135645X.

[34] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[35] S. P. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2): 129–137, 1982.

[36] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, pages 103–130, 1997.

[37] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[38] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*, pages 41–48. AAAI Press, 1998.

[39] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[40] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142. Springer-Verlag, 1998.

[41] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.

[42] C. C. Aggarwal. *Data Mining*. Springer International Publishing, 2015.

[43] K.-B. Duan and S. S. Keerthi. Which is the best multiclass svm method? an empirical study. pages 278–285. Springer-Verlag, 2005.

[44] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.*, 13(2):415–425, 2002.

[45] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49. ACM, 1999.

[46] K. Aas and L. Eikvil. Text categorisation: A survey. *Raport NR*, 941, 1999.

[47] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *In Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[48] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168. ACM, 2006.

[49] Y. Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, pages 69–90, 1999.

[50] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2, 2006.

[51] J. Wang, P. Neskovic, and L. N. Cooper. Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recogn. Lett.*, 28(2):207–213, 2007. ISSN 0167-8655.

[52] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[53] S. Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Syst. Appl.*, 28 (4):667–671, May 2005.

[54] G. Valentini and F. Masulli. Ensembles of learning machines. In *Proceedings of the 13th Italian Workshop on Neural Nets-Revised Papers*, pages 3–22. Springer-Verlag, 2002.

[55] T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000.

[56] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–297. ACM, 1996.

[57] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 279–287. ACM, 1996.

[58] S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

[59] W. Lam and K.-Y. Lai. A meta-learning approach for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 303–309. ACM, 2001.

[60] P. N. Bennett, S. T. Dumais, and E. Horvitz. Probabilistic combination of text classifiers using reliability indicators: Models and results. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 207–214. ACM, 2002.

[61] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 215–223. ACM, 1998.

[62] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. In *Machine Learning*, pages 135–168, 2000.

[63] Y.-H. Kim, S.-Y. Hahn, and B.-T. Zhang. Text filtering by boosting naive bayes classifiers. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 168–175. ACM, 2000.

[64] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 334–343. ACM, 1997.

[65] L. Breiman. Pasting small votes for classification in large databases and on-line. *Mach. Learn.*, 36 (1-2):85–103, 1999.

[66] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[67] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40(2):139–157, 2000.

[68] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 86–93. ACM, 2000.

[69] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers Inc., 1999.

[70] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, 2000.

[71] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 19–26. Morgan Kaufmann Publishers Inc., 2001.

[72] X. Zhu and Z. Ghahramani. Towards semi-supervised classification with markov random fields, 2002.

[73] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, Dec. 2006.

[74] X. Zhu. Semi-supervised learning literature survey, 2006.

[75] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2004.

[76] P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[77] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[78] D. D. Lewis. Evaluating text categorization. In *In Proceedings of Speech and Natural Language Workshop*, pages 312–318. Morgan Kaufmann, 1991.

[79] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995.

[80] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *J. Intell. Inf. Syst.*, 18(2-3):219–241, 2002.

Appendix A

Portuguese Codebook and DRE Corpus Frequencies

Table A.1: Minor Categories and document frequencies.

Categories	Sub-Categories	NºDocs
Macroeconomia	Geral	4
	Política monetária, Banco Central e tesouro	8
	Orçamento de Estado e Dívida pública	22
	Política fiscal, impostos e reforma fiscal (IVA)	59
	Política industrial	8
	Controlo e estabilização de preços	1
	Outros	3
Direitos e Liberdades civis e das minorias		
	Discriminação por género e orientação sexual	2
Saude	Geral	1
	Serviço Nacional de Saúde	1
	Seguros, Sistemas Alternativos de Saúde e Custos	1
	Regulação da industria e do comércio farmacêutico e dos laboratórios clínicos	2
	Hospitais e manutenção de infra-estruturas	11
	Acordos com entidades privadas; regulação; reembolsos	2
	Gestão e Regulação de Recursos humanos	10
	Prevenção e promoção da saúde	1
	Saúde infantil e pré-natal	1
	Cuidados continuados, doenças terminais e reabilitação	8
	Despesa com medicamentos, tabelas de comparticipação e custos	1
	Tabagismo, Tratamento e Educação	1
	Outros	1
Agricultura, Pecuária e Pescas	Geral	3
	Comércio agrícola e pecuário	3
	Subsídios à agricultura e seguros	2
	Segurança e controlo alimentar	2
	Promoção e divulgação agrícola	1
	Controlo de doenças e pragas e bem-estar animal	3
	Pesca e Caça	9
	Pesquisa e desenvolvimento agrícola	1
	Outros	2
Trabalho e emprego	Geral	5
	Segurança e proteção no trabalho	5
	Formação e desenvolvimento profissional	1
	Relações com sindicatos	2
	Regulação das relações laborais	1
	Emprego jovem e mão de obra infantil	2
Educaçao	Geral	1
	Ensino superior	91
	Ensino básico e secundário	22
	Ensino Profissional	9
	Outros	6
Ambiente	Geral	4
	Potabilidade da água, Abastecimento de Água, Poluição da Água e Conservação	2
	Poluição do ar, aquecimento global e poluição sonora	1
	Proteção de espécies, florestas e caça	4
	Outros	1
Energia	Eletricidade e energia hidroelétrica	3
	Gás natural e petróleo	3
	Eficiência energética	4
Migrações		
Transportes	Geral	9
	Transportes públicos e segurança	19
	Veículos ligeiros e pesados, trânsito e segurança, construção, manutenção e segurança de estradas	16
	Aeroportos, linhas aéreas, controlo e segurança de tráfico aéreo	25
	Transporte ferroviário e segurança	2
	Transportes marítimos fluviais e Indústria Naval	43
	Obras públicas	2
	Outros	3
Justiça e Direito	Geral	6
	Instituições de combate ao crime	37
	Controlo do tráfico, produção e consumo de drogas	1
	Sistema judicial	24
	Sistema prisional	4
	Proteção civil	32
	Código civil e penal	7
Politicas Sociais	Geral	1
	Apoio à terceira idade	6
	Apoio a pessoas com deficiência e doenças crónicas	6
	Serviços sociais e voluntariado	3
	Apoio à família e à infância	6
	Regimes Complementares de Segurança Social	5
	Outros	2
Habitação	Geral	11
	Políticas de Habitação e desenvolvimento da comunidade	48
	Desenvolvimento rural	1
	Habitação social	7
	Outros	1
Sector financeiro, industria e comercio	Geral	5
	Sistema bancário e regulação financeira	9
	Mercado de Valores	9
	Regulação de seguros	9
	Pequenas e Médias Empresas	10
	Direitos de autor e patentes	3
	Desastres naturais	10
	Turismo	8
	Defesa do consumidor	4
	Desporto, Lazer e Regulação do Jogo	3
	Ordens profissionais e associações comerciais científicas	1
	Outros	15
Defesa	Geral	3
	Forças armadas	9
	Carreiras militares	30
	Aquisição de armamento e material militar	1
	Instalações militares	2
	Emprego civil nas instalações militares	1
	Operações de Guerra	2
	Outros	1
Ciencia, tecnologia e comunicaçoes	Regulação e infraestruturas de Telecomunicações	22
	Regulação e Infraestruturas da Comunicação social	11
Comercio externo	Geral	2
	Acordos de comércio externo	4
	Investimento estrangeiro	1
	Regulação de Importações	3
	Câmbios	1
Politica externa	Cooperação internacional	2
	Europa Ocidental e União Europeia	3
	Outros Países	2
	Organizações Internacionais (não financeiras) ; Organizações não governamentais Internacionais	4
	Diplomacia e diplomatas	23
Governo e administraçao publica	Geral	6
	Relações entre níveis de governo	8
	Eficiência governativa	339
	Correios	9
	Administração Pública	197
	Nomeações e exonerações	35
	Condecorações, reconhecimento público, fabrico e cunhagem de notas e moedas	119
	Compras Públicas	46
	Gestão do setor público	21
	Administração fiscal	25
	Funcionamento do Parlamento e relação com o Governo e Presidência da República	1
	Campanhas Políticas, eleições e Partidos Políticos	4
	Transição e consolidação da democracia	15
	Descolonização e independências das antigas colónias	1
	Outros	2
Recursos naturais e Património	Geral	1
	Parques Naturais, Reservas e Património cultural	2
	Utilização de recursos naturais, Minas, Florestas	1
	Recursos Hídricos: desenvolvimento, obras públicas	7
Política cultural		

Appendix B

Other Collected Datasets

B.1 Foreign Datasets

Three datasets with similar codebook classifications, out of the scope of this work, are presented here.

B.1.1 French Corpus

The French Comparative Policies Agendas group makes available online[1] several datasets with a coding style similar to the Comparative Agendas categories coding style. From the provided datasets, the "*The laws of the 5th Republic*" was selected to be used as a comparative measure dataset.

The full document integral texts of the legislation are not available, and collecting it involves a big effort in parsing and database creation. Therefore only the descriptions of the documents will be used as documents in this dataset, similarly to the methodology employed in the DRE-META dataset. It will be simply referred to as FR corpus. The goal will be analysing the discriminative power of the small feature set which has also skewed classes but in contrary to the DRE corpus it has the documents more evenly distributed by the categories as can be seen in the summary of the major categories in table B.2.

As one should expect the number of features in this corpus will be smaller, which can be seen in the summary of the corpus characteristics in table B.1. Similarly to the DRE-META corpus, it is likely that the information contained in it will be condensed and concise. In figure B.1 the distribution of features in the documents is considerably less than the mean for almost half of the corpus which can arise doubts if some documents contain information relevant enough to be correctly classified.

Table B.1: FR, ES and EUA corpora statistics.

Corpus	Total Words	Unique Words	Mean words per doc
FR	54184	3801	18.293
ES	26521	2689	16.957
USA	174714	11825	8.773

[1] French Policy Agendas `http://www.agendas-france.fr/`

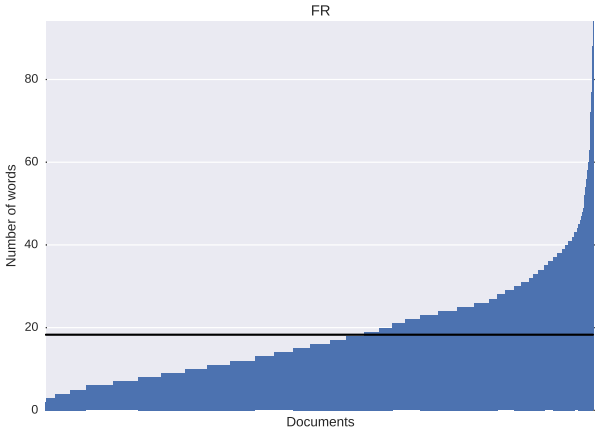


Figure B.1: FR Document length distributions. The plotted black line corresponds to the mean document length.

Table B.2: Major Categories and document frequencies of FR Corpus.

Categories	NºDocs
Politique Macroéconomique	191
Droits de l'homme, libertés publiques et discriminations	95
Santé	82
Agriculture, pêche et sylviculture	70
Travail et emploi	207
Education	49
Environnement	80
Energie	27
Immigration	73
Transport	152
Justice et Criminalité	346
Politique sociale	44
Politiques urbaines et territoriales	73
Régulations économiques	187
Défense	107
Espace, Science, Technologie et communication	57
Commerce extérieur	128
Affaires internationales et aide extérieure	493
Fonctionnement de l'Etat	379
Domaine public et gestion de l'eau	14
Politique culturelle	86
Sports	22

B.1.2 Spanish Corpus

The Spanish Policy Agendas[2] also provides some coded datasets, without the integral text as well. In the same fashion as the DRE-META and FR corpora, the "*Laws*" dataset descriptions of the coded documents will be used as documents, which to this corpus it will be referred to as ES corpus.

In can be seen in table B.1 and figure B.2, this corpus has a similar distribution to the FR corpus but with fewer documents in the collection, and consequently a less diverse feature set. It also shows some class distribution unbalance.

B.1.3 USA Corpus

The Congressional Bills Project[3] provides datasets with bills published in the United States congresses. The dataset chosen corresponds to the bills of "*93rd through 113th*" congresses. This corpus will be

[2]Spanish Comparative Agendas `http://www.ub.edu/spanishpolicyagendas/`

Figure B.2: ES Document lengths distributions. The plotted black line corresponds to the mean document length.

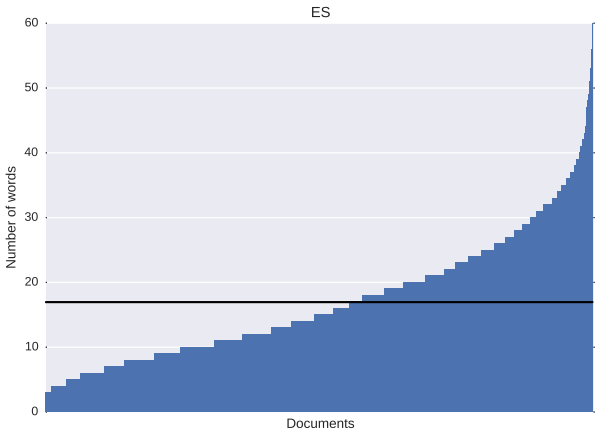


Table B.3: Major Categories and document frequencies of ES Corpus.

Categories	NºDocs
Macroeconomía	255
Derechos, libertades civiles, problemas relativos a las minorías	40
Salud	40
Agricultura e industria pesquera	63
Trabajo	90
Educación y cultura	64
Medio Ambiente	40
Energía	33
Transporte	92
Política interior y justicia	170
Política social	21
Planificación urbanística y política de vivienda	19
Política industrial y comercio	191
Política de defensa	62
Investigación, tecnología y comunicaciones	46
Comercio exterior	11
Política Exterior	79
Gobierno y Administración Pública	192
Recursos naturales y gestión del agua	56

simply named USA corpus.

The USA corpus, as the DRE-META, ES and FR corpora, was created using only the short descriptions provided in the coded datasets. As can be seen in table B.1 this dataset has a much lower mean of words per document than the other presented datasets, however due to being a larger collection of coded documents the number of unique words in the feature set ends up being richer than the previous three corpus which were created using the same methodology. The document class distribution can be found in table B.4 and the document length distribution in figure B.3.

The motivation for using such corpus, besides the same reasons stated for the FR corpus, is to evaluate if the classification with such small feature set per document can, in overall, be compensated by the large number of samples.

[3]Congressional Bills Project `http://congressionalbills.org/index.html`

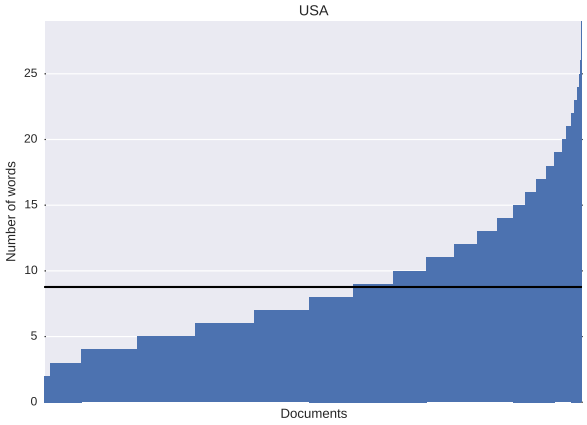


Figure B.3: USA Document lengths distributions. The plotted black line corresponds to the mean document length.

Table B.4: Major Categories and document frequencies of USA Corpus.

Categories	**NºDocs**
Macroeconomics	409
Civil Rights, Minority Issues, and Civil Liberties	143
Health	819
Agriculture	737
Labor, Employment, and Immigration	266
Education	492
Environment	518
Energy	356
Immigration	157
Transportation	1185
Law, Crime, and Family Issues	853
Social Welfare	292
Community Development and Housing Issues	264
Banking, Finance, and Domestic Commerce	892
Defense	2179
Space, Science, Technology and Communications	386
Foreign Trade	517
International Affairs and Foreign Aid	835
Government Operations	4448
Public Lands and Water Management	4166

B.2 Suppervised Experiments with other Datasets

Using only the descriptions of the laws of the ES, FS and EUA corpora, here it is presented the results of the same experiments performed to the DRE corpora, with exception of the feature transformation with LSI experiment since it is a method which aims to capture latent semantics in the text, and in these datasets there is no reasonable amount of text that justifies such task.

B.2.1 Baseline and Weighting Impact Evaluation

The FR, ES and USA corpus, even though they have less features per document, overall the classification showed to be considerably good. A special remark to the $F1_M$ scores of the USA corpus, which indicates that a corpus with classes well populated provides enough information to classify reasonably good all classes. The results can be found in table B.5

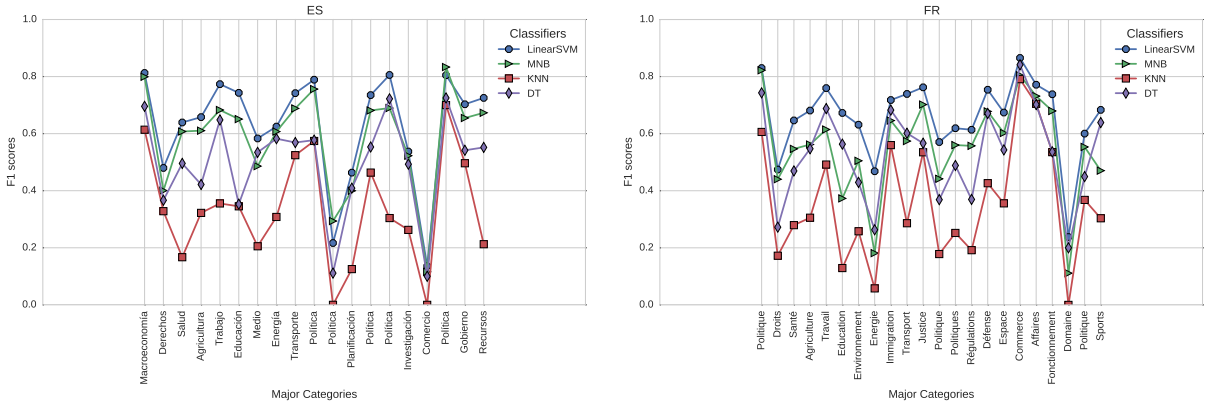
The DT classifier is the worst in terms of training execution time due to the heavy cost of constructing and storing the necessary classification tree. In the USA corpus it can be seen that the implementation scales badly for large datasets. The computation times can be seen in figure B.5.

Table B.5: Impact of Feature Weighting in ES,FR and USA corpora.

		LinearSVM		MNB		KNN		DT	
Corpus	Weighting	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$
FR	TF	0.728	0.660	0.646	0.556	0.479	0.356	0.587	0.525
	Norm(TF)	0.727	0.662	0.652	0.546	0.521	0.399	0.537	0.477
	TFIDF	0.735	0.671	0.664	0.579	0.511	0.431	0.583	0.525
	Norm(TFIDF)	0.739	0.684	0.683	0.598	0.637	0.557	0.540	0.488
	$TFIDF_{log}$	0.740	0.680	0.677	0.584	0.514	0.431	0.584	0.530
	$Norm(TFIDF_{log})$	0.746	0.691	0.682	0.596	0.636	0.555	0.532	0.487
ES	TF	0.721	0.627	0.676	0.582	0.458	0.331	0.578	0.499
	Norm(TF)	0.729	0.642	0.696	0.592	0.504	0.372	0.504	0.431
	TFIDF	0.742	0.661	0.700	0.594	0.530	0.423	0.569	0.491
	Norm(TFIDF)	0.750	0.670	0.709	0.609	0.643	0.522	0.505	0.435
	$TFIDF_{log}$	0.744	0.666	0.693	0.597	0.530	0.416	0.574	0.498
	$Norm(TFIDF_{log})$	0.740	0.654	0.706	0.605	0.655	0.532	0.514	0.444
USA	TF	0.802	0.756	0.743	0.690	0.655	0.575	0.720	0.675
	Norm(TF)	0.801	0.757	0.752	0.697	0.717	0.641	0.701	0.652
	TFIDF	0.803	0.760	0.749	0.700	0.671	0.615	0.719	0.674
	Norm(TFIDF)	0.807	0.767	0.748	0.701	0.743	0.692	0.696	0.652
	$TFIDF_{log}$	0.804	0.761	0.748	0.698	0.671	0.616	0.718	0.675
	$Norm(TFIDF_{log})$	0.808	0.768	0.747	0.700	0.743	0.692	0.697	0.655

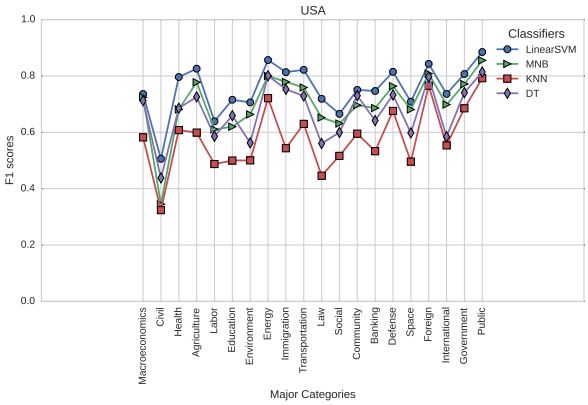
B.2.2 Stemming Evaluation

Datasets with lower dimensional spaces as the seem to benefit from stemming. This could be explained by the principal purpose of the stemming which is mapping different words into their root feature (word), benefiting by having more features represented in different documents. The results are summarized in table B.6.



(a) ES

(b) FR



(c) USA

Figure B.4: F1 scores for each category of the ES, FR and EUA corpora and classifier with features weighted with $||TFIDF||$.

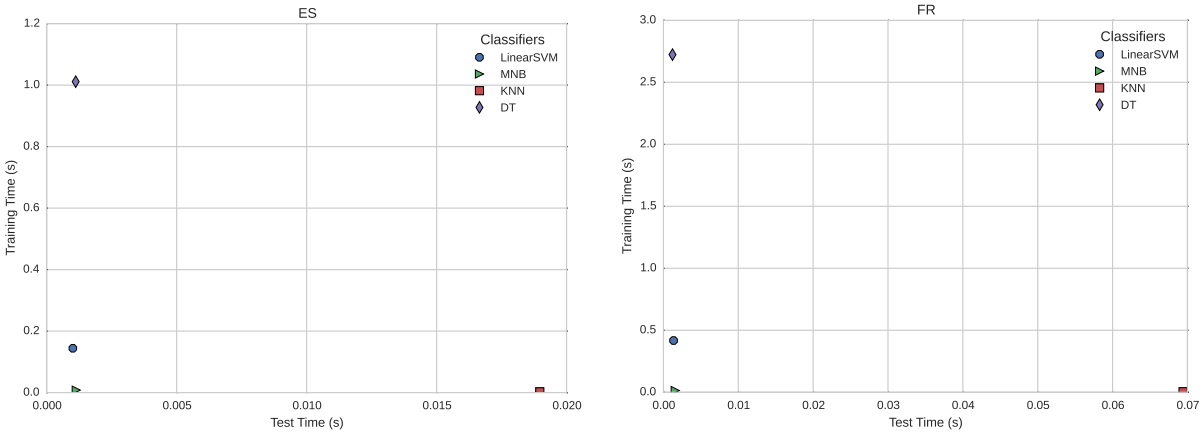
The stemming process for the FR corpus resulted in a feature space of 2669 unique features, the ES corpus with a reduction of 31% and 1854 features, and finally the USA corpus with a reduction of 27% with 8597 unique features.

Table B.6: Impact of Feature Stemming in the FR, ES and EUA corpora.

	LinearSVM		MNB		KNN		DT	
Corpus	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$	$F1_m$	$F1_M$
FR	0.722	0.661	0.654	0.563	0.482	0.363	0.586	0.526
FR Stemmed	0.733	0.673	0.651	0.571	0.482	0.356	0.622	0.566
ES	0.720	0.627	0.686	0.592	0.454	0.333	0.571	0.497
ES Stemmed	0.743	0.660	0.690	0.607	0.473	0.348	0.615	0.541
USA	0.800	0.754	0.757	0.701	0.654	0.577	0.718	0.675
USA Stemmed	0.800	0.756	0.755	0.699	0.664	0.588	0.718	0.671

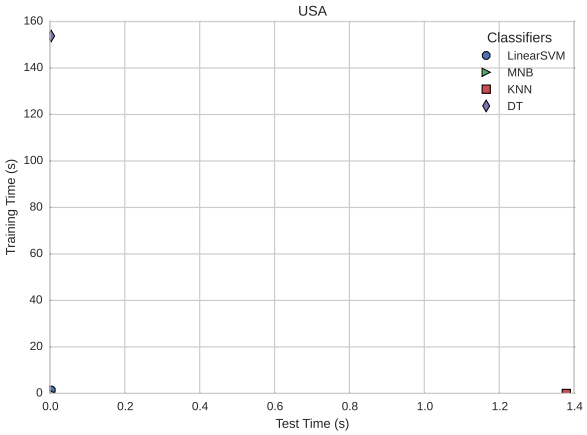
B.2.3 Impact of the Training Size in Classification Performances

Similarly as seen in the DRE corpora experiments, as the training size grows, the classification accuracy increases. The impact is lower in the USA corpus, since it contains a big set of classified documents, resulting in higher scores for lower percentage of the set. Classification results can be found in table B.7 and figure B.6.



(a) ES

(b) FR



(c) USA

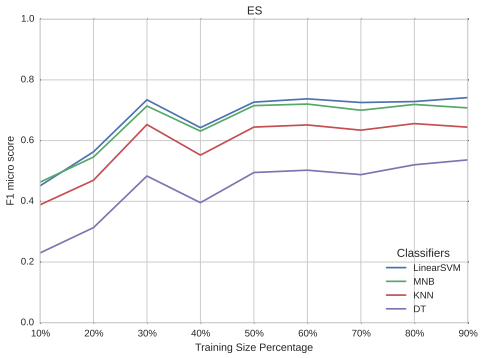
Figure B.5: Computation time for training and testing for multiple classifiers for the FR, ES and EUA corpora.

B.2.4 Evaluation of Feature Selection Methods

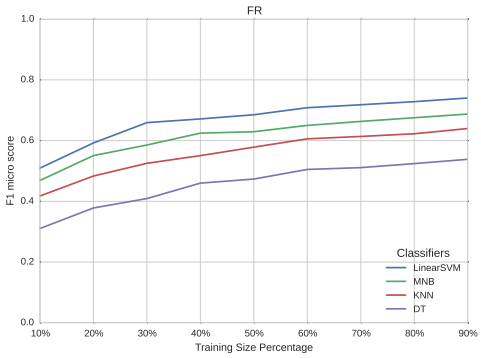
χ^2 feature selection method was the best metric for feature selection in these corpora as well. With only 20% of the most significant feature set for the USA corpus, while in the case of ES and FR, 40% of the feature set was necessary to achieve similar classification scores of the original set. Classification results can be found in table B.8 and figure B.7. Computation time for the feature selection metrics in figure **??**.

Table B.7: Impact of the multiple train size in classification performances for the FR, ES and EUA corpora.

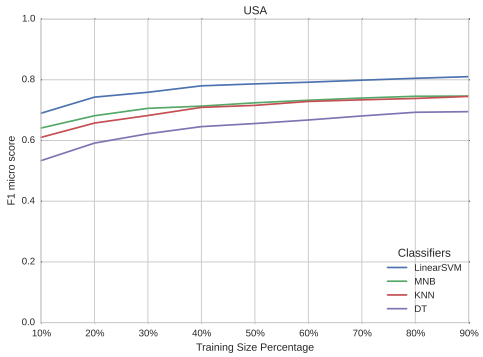
Corpus	Classifiers	avg	Training Sizes 10%	20%	30%	40%	50%	60%	70%	80%	90%
FR	LinearSVM	F_M	0.403	0.502	0.566	0.604	0.625	0.655	0.656	0.664	0.694
		F_m	0.509	0.592	0.659	0.671	0.685	0.708	0.718	0.728	0.740
	MNB	F_M	0.343	0.444	0.478	0.517	0.525	0.572	0.580	0.593	0.606
		F_m	0.469	0.551	0.586	0.625	0.629	0.650	0.663	0.675	0.688
	KNN	F_M	0.279	0.366	0.412	0.447	0.469	0.514	0.524	0.527	0.564
		F_m	0.418	0.483	0.525	0.551	0.578	0.606	0.614	0.623	0.640
	DT	F_M	0.232	0.314	0.350	0.407	0.409	0.461	0.472	0.460	0.482
		F_m	0.310	0.378	0.409	0.460	0.473	0.505	0.511	0.524	0.538
ES	LinearSVM	F_M	0.339	0.439	0.638	0.535	0.637	0.654	0.644	0.644	0.661
		F_m	0.451	0.563	0.734	0.643	0.727	0.738	0.725	0.729	0.742
	MNB	F_M	0.310	0.420	0.621	0.497	0.631	0.637	0.605	0.629	0.613
		F_m	0.463	0.546	0.714	0.631	0.715	0.720	0.700	0.719	0.708
	KNN	F_M	0.248	0.330	0.532	0.417	0.526	0.531	0.511	0.534	0.514
		F_m	0.389	0.469	0.653	0.552	0.645	0.652	0.634	0.656	0.644
	DT	F_M	0.176	0.253	0.410	0.319	0.436	0.427	0.420	0.439	0.456
		F_m	0.230	0.313	0.483	0.396	0.495	0.503	0.488	0.520	0.537
USA	LinearSVM	F_M	0.619	0.683	0.701	0.726	0.740	0.747	0.757	0.759	0.772
		F_m	0.690	0.743	0.759	0.780	0.786	0.792	0.799	0.805	0.810
	MNB	F_M	0.545	0.600	0.637	0.648	0.668	0.681	0.691	0.694	0.698
		F_m	0.641	0.681	0.706	0.713	0.724	0.732	0.740	0.746	0.746
	KNN	F_M	0.516	0.579	0.610	0.646	0.657	0.674	0.678	0.681	0.694
		F_m	0.610	0.657	0.682	0.709	0.716	0.729	0.734	0.738	0.745
	DT	F_M	0.469	0.535	0.568	0.595	0.608	0.622	0.634	0.649	0.648
		F_m	0.534	0.591	0.622	0.646	0.656	0.667	0.681	0.693	0.695



(a) ES



(b) FR

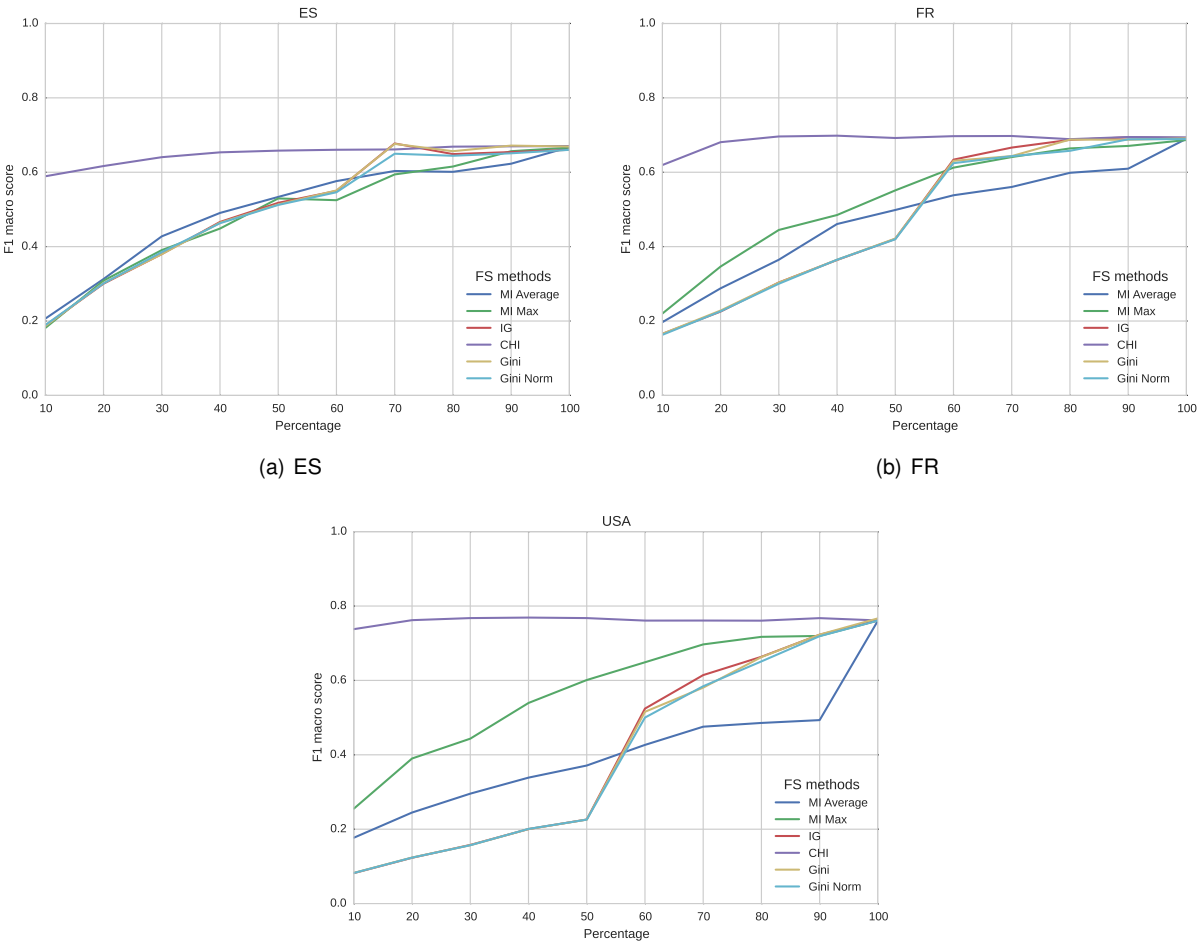


(c) USA

Figure B.6: F1-micro scores for the FR, ES and USA corpora and classifiers with multiple train sizes.

Table B.8: Multiple feature selection methods for the SVM classifier with the FR, ES and EUA corpora.

			Feature Percentage									
Corpus	Methods	avg	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
FR	MI Average	F_M	0.197	0.288	0.365	0.461	0.499	0.538	0.560	0.599	0.610	0.690
		F_m	0.222	0.272	0.345	0.439	0.469	0.522	0.557	0.606	0.638	0.741
	MI Max	F_M	0.220	0.346	0.445	0.485	0.551	0.612	0.641	0.664	0.671	0.687
		F_m	0.224	0.287	0.379	0.421	0.500	0.592	0.648	0.698	0.715	0.739
	IG	F_M	0.165	0.225	0.304	0.364	0.421	0.634	0.666	0.687	0.694	0.692
		F_m	0.222	0.266	0.309	0.316	0.388	0.630	0.696	0.718	0.736	0.746
	CHI	F_M	0.619	0.681	0.696	0.698	0.692	0.697	0.697	0.689	0.695	0.694
		F_m	0.637	0.718	0.738	0.749	0.745	0.750	0.752	0.741	0.748	0.746
	Gini	F_M	0.166	0.228	0.303	0.365	0.422	0.631	0.643	0.688	0.687	0.691
		F_m	0.223	0.268	0.310	0.322	0.404	0.627	0.658	0.720	0.728	0.748
	Gini Norm	F_M	0.163	0.226	0.300	0.365	0.420	0.624	0.643	0.658	0.689	0.689
		F_m	0.222	0.267	0.307	0.336	0.391	0.564	0.623	0.666	0.724	0.741
ES	MI Average	F_M	0.206	0.313	0.428	0.491	0.534	0.576	0.603	0.601	0.623	0.668
		F_m	0.260	0.344	0.451	0.513	0.556	0.604	0.631	0.641	0.673	0.750
	MI Max	F_M	0.181	0.309	0.391	0.449	0.530	0.525	0.594	0.615	0.657	0.666
		F_m	0.212	0.281	0.340	0.403	0.493	0.517	0.611	0.644	0.728	0.750
	IG	F_M	0.187	0.300	0.380	0.466	0.518	0.550	0.677	0.649	0.654	0.660
		F_m	0.264	0.344	0.401	0.456	0.517	0.547	0.714	0.713	0.735	0.744
	CHI	F_M	0.589	0.617	0.641	0.654	0.658	0.660	0.661	0.669	0.670	0.670
		F_m	0.623	0.682	0.707	0.728	0.738	0.741	0.743	0.754	0.753	0.751
	Gini	F_M	0.186	0.302	0.380	0.464	0.513	0.551	0.676	0.657	0.672	0.669
		F_m	0.262	0.343	0.400	0.455	0.516	0.546	0.717	0.719	0.746	0.751
	Gini Norm	F_M	0.189	0.301	0.385	0.463	0.512	0.546	0.650	0.644	0.651	0.660
		F_m	0.263	0.344	0.403	0.453	0.515	0.546	0.657	0.687	0.721	0.744
USA	MI Average	F_M	0.177	0.245	0.296	0.339	0.371	0.427	0.476	0.486	0.493	0.761
		F_m	0.242	0.272	0.305	0.335	0.364	0.431	0.485	0.508	0.521	0.801
	MI Max	F_M	0.255	0.390	0.443	0.539	0.601	0.649	0.697	0.717	0.720	0.761
		F_m	0.279	0.338	0.371	0.456	0.529	0.596	0.678	0.717	0.721	0.801
	IG	F_M	0.082	0.123	0.158	0.200	0.226	0.524	0.614	0.664	0.723	0.761
		F_m	0.269	0.297	0.319	0.341	0.364	0.550	0.654	0.721	0.772	0.801
	CHI	F_M	0.738	0.762	0.768	0.769	0.768	0.761	0.761	0.761	0.767	0.761
		F_m	0.776	0.799	0.804	0.807	0.807	0.802	0.801	0.801	0.808	0.801
	Gini	F_M	0.082	0.124	0.157	0.200	0.226	0.516	0.581	0.663	0.723	0.767
		F_m	0.269	0.298	0.319	0.341	0.363	0.537	0.634	0.720	0.775	0.806
	Gini Norm	F_M	0.082	0.124	0.157	0.201	0.226	0.500	0.584	0.651	0.719	0.761
		F_m	0.269	0.297	0.319	0.341	0.363	0.492	0.531	0.655	0.753	0.801



(a) ES

(b) FR

(c) USA

Figure B.7: F1 micro scores for each feature selection methods for the ES, FR and USA corpora with SVM.