TÉCNICO LISBOA

Automatic Correspondence Distribution for a Public Institution

André Miguel Balau Fazendeiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Ricardo Daniel Santos Faro Marques Ribeiro
Prof. Nuno João Neves Mamede

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Ricardo Daniel Santos Faro Marques Ribeiro
Member of the Committee: Prof. Fernando Manuel Marques Batista

January 2021

Acknowledgments

This section aims to manifest my gratitude to all the people who offered me support during this dissertation and without whom it certainly wouldn't be possible. I shall address to them in Portuguese:

Em primeiro lugar, gostaria de agradecer à minha família. Ao meu pai, mãe e irmão, agradeço a paciência, amor e apoio que sempre me providenciaram e sem os quais nada disto seria imaginável.

Agradeço a todo o pessoal docente que me acompanhou no meu percurso académico, em especial aos meus orientadores, Professor Ricardo Ribeiro e Professor Nuno Mamede. São incontáveis os conselhos valiosos que me deram e as chamadas de atenção nos momentos necessários.

Agradeço também aos meus colegas do HLT@INESC-ID, em especial à Vilma Neves pelas dúvidas partilhadas e esclarecimentos prestados.

À minha namorada, agradeço a motivação e confiança que depositou em mim, no decorrer de toda esta etapa.

Por último, agradeço a todos os meus amigos, por todos os momentos de companheirismo e amizade que por um ou outro motivo tornaram este trabalho uma experiência ainda mais recompensadora.

A todos eles, obrigado por toda a paciência, apoio e afeto que depositaram em mim.

Resumo

A distribuição de correspondência é de extrema importância numa grande organização como a Marinha Portuguesa. Um documento mal encaminhado pode ter graves repercussões, tais como a não realização de tarefas importantes e a perda de informação.

Com o passar do tempo, a classificação do texto evoluiu de modelos de frequência de palavras para modelos sequenciais com *word embeddings*. Esta mudança de paradigma é atualmente o estado da arte e revela resultados promissores em datasets de grande escala.

Atualmente, a correspondência dentro da Marinha é classificada à mão, tarefa morosa que pode ser propensa a erro humano. Assim, esta dissertação aborda este problema, estudando alternativas viáveis para a classificação automática de textos, através de Machine Learning e ferramentas de Processamento de Linguagem Natural.

Com este objectivo em mente, vários modelos de Machine Learning foram testados e estudados, alguns deles mostrando resultados positivos, tais como Regressão Logística, com mais de 90% de acurácia média em todas as etiquetas e um *exact match ratio* de aproximadamente 50%.

Palavras-chave: Distribuição de Correspondência, Classificação Multi-label, Processamento de Linguagem Natural, Machine Learning

Abstract

Correspondence distribution is of utter importance in a large organization such as the Portuguese Navy. A misdirected document might have severe repercussions such as important tasks not being performed and information being lost.

Over time, text classification went from relying solely on word frequency models to sequential models with word embeddings. This paradigm shift is currently the state of the art and reveals promising results in large scale datasets.

Currently, correspondence within the Navy is classified by hand which can be prone to human error and time-consuming. Hence, this dissertation addresses this problem, studying viable alternatives for automatic text classification, relying on Machine Learning and Natural Language Processing tools.

With this goal in mind, various machine learning models were tested and studied, with some of them showing positive results, such as Logistic Regression, with over 90% average accuracy over all labels and an average Exact Match Ratio of approximately 50%.

Keywords: Correspondence Distribution, Multi-label Classification, Natural Language Processing, Machine Learning

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In a world with an ever growing data flux, large organizations demand fast and efficient information management. The lack of a proper system to categorize, store and retrieve documents may result in lost information, which can be highly detrimental to any company's productivity. For this reason, computer systems are designed and used to provide the power and accessibility of fast indexation and document security.

Regarding the Portuguese Navy, storage and distribution of correspondence is currently done by hand in a somewhat outdated manner that increases the risk for human error and subjectivity underlying document classification which may lead to lost documents and difficult retrievals.

Text classification is an essential subject of Machine Learning and Natural Language Processing that makes possible the distinction and grouping of documents by a machine, based on the written text. Text classification has many use cases linked to automation and efficiency. The importance of this area is linked to the fact that tasks which can be performed now by text classification algorithms had to be done previously through human labor, with higher costs in terms of efficiency.

Multi-label Classification is the classification task that works specifically with data that has a set of outputs (e.g.: $\{A\}$, $\{B\}$, or $\{A,B\}$), rather than a binary (e.g.: A or B) or a multi-class problem which has a single output chosen from multiple classes (e.g.: A, B or C). This type of classification involves mainly two approaches to the problem, being them Problem Transformation and Algorithm Adaptation, and introduces distinct metrics for evaluating performance which will be addressed later.

The purpose of this work is to propose efficient methods for Multi-label Classification in text documents, in order to automate the distribution of documents within the Portuguese Navy and to test their reliability in the proposed scenario. The task to accomplish is, given a text document, to accurately choose within the possible recipients, to whom the message should be forwarded and what degree of action is required from the recipients – whether the message is for information or actions must be taken.

1.1 Motivation

Nowadays data flows at an increasingly fast pace. Companies need to organize huge datasets in order to store them in an accessible manner. Through the last years, Machine Learning and Natural Language Processing have progressed at a steady pace to help in this field of information management, classification and retrieval. Multiple distinct classifiers have shown promise in this domain and techniques such as sequential models with word embeddings have made significant contributions in the area [1, 2].

The motivation to work on this dissertation is to experiment with such methods while working on a relevant real world problem, aiming to achieve a reliable faster method that can replace or complement the policies currently in use by the Portuguese Navy.

As for the Portuguese Navy the outcome of this dissertation might pave the way for a computer system that can reduce workload for the responsible party in charge of the classification and optimize the distribution of correspondence and retrieval of documents when needed, proving beneficial for the organization.

The developed model can be also adjusted to work in other document tagging problems, with multiple degree tags, similar to the dataset in study.

1.2 Problem Description

The focus of this work is to develop a model that when provided with a document, can correctly identify the recipients and to what degree an action is required from them.

Each document sent across the Portuguese Navy possesses a table (such as the one presented in Figure 1.1) with at least seven departments within the Navy.

Distribuição	
STI	
CDIACM	
DAGI	
DITIC	C
C/GAB	
SERV. PART.	C
ADJ. SEC1	

Figure 1.1: A representation of the graphical correspondence distribution table present in the Portuguese Navy documents.

The seven existing departments present in every document's distribution table are described as follows:

- STI - *Superintendência das Tecnologias da Informação*
- CDIACM - *Centro de Documentação de Informação e Arquivo Central da Marinha*
- DAGI - *Direção Análise e Gestão de Informação*

- DITIC - *Direção de Tecnologias de Informação e Comunicações*
- CGAB - *Gabinete do Superintendente das TI*
- SERV.PART - *Serviço Particular*
- ADJ.SEC1 - *Entidade Contabilística*

Other three departments also appear in the dataset, but in a residual manner, being therefore discarded from the dataset.

For each of the listed departments, a character is assigned as a descriptor to the degree of action required from that specific department:

- Blank (0) - if no action is required and that department is not a recipient
- Letter C - if the document in question is for information (*Conhecimento*) but no action is required from said department
- Letter A - if the document is to be addressed to said department and the latter is required to take action (*Ação*)

Looking into the example from Figure 1.1, the document will be distributed to departments number 3 and 6 and since the label for both departments is *C* (*Conhecimento* - Information) no direct action needs to be taken by those departments.

The purpose of this dissertation is to study the viability of implementing this classification as an automatic process. This meaning that when provided with a document, the proposed solution should output an accurate distribution table, predicting correctly the action degree needed for each department.

In other words: for each provided document, our model should output the accurate class (*blank*, *A* or *C*) for each label (department) in the label set: *STI*, *CDIACM*, *DAGI*, *DITIC*, *CGAB*, *SERV.PART* and *ADJ.SEC1*.

This problem fits within the Multi-label Classification task, where for each sample the set of correct labels must be returned. For this task in specific, the correct class corresponding to the specific action degree must be returned for each label.

Related work focus in adapting the data to work with classical classifiers (Problem Transformation) or adapting the algorithms to specifically work with Multi-label data (Algorithm Adaptation). Both methods shall be discussed more in-depth in Chapter 2.

1.3 Objectives

This work addresses the task of Multi-Label Classification applied to the environment of the Portuguese Navy. The aim is to study state of the art text classification models and to test and implement a model that is less time consuming than the manual alternative and provides similar or better accuracy, proving to be a reliable alternative to be adopted.

With that in mind, several classifiers were surveyed using distinct classification models and text representations. Regarding their performance the results will be compared to similar methods from the state of art and against all models implemented, since it heavily depends in the available data and chosen methods.

1.4 Contributions

This dissertation studied the use of Binary Relevance models in order to correctly predict the classification for seven labels regarding a written document. With this in mind, nine models were implemented and their performance compared with each other and analyzed.

Besides this and attending to the imbalanced nature of our dataset, 3 distinct balancing methods were experimented in order to find a reliable method with as little bias as possible.

This work poses itself as part of a bigger project, which aims to develop a data management software for the Portuguese Navy, which will improve efficiency in the organization and reduce the influence of human error.

1.5 Thesis Outline

- Chapter 2 - Background

  Provides contextualization on the themes addressed by this dissertation and provides a look at the related work currently available.

- Chapter 3 - A Classifier System for Correspondence Distribution

  Describes the process of building and testing a machine learning model tailored for Multi-label Text Classification.

- Chapter 4 - Results

  Focuses on the results achieved by the implemented models, together with a possible explanation for said results.

- Chapter 5 - Conclusions

  Summarizes the outcome of this dissertation and proposes work that may be done in the future to supplement this work.

Chapter 2

Background

The purpose of this chapter is to level the reader with the work described in this document. Section 2.1 discusses some of the fundamental concepts necessary to understand this dissertation. Section 2.2 explains briefly the hierarchy of the Portuguese Navy's units, that are referenced in the dataset. Section 2.3 expands on other articles related to the case under study.

2.1 Fundamental Concepts

This section briefly describes both classification issues inherent to this dissertation, being them Multi-label Classification and Text Classification.

2.1.1 Multi-Label Classification

Multi-Label Classification is the task of selecting from within a list of possible labels, the ones which should be associated with the object we are trying to classify. Multi-Label Classification has multiple use cases, like text and sound categorization [3, 4], semantic scene classification [5], medical diagnosis [6, 7] or gene and protein function classification [8].

Multi-Label Classification differs from classification tasks like Binary Classification and Multi-Class Classification, as the first does not take into consideration dependencies between distinct labels and in the latter, it is only attributed one label per document [9], which does not accomplish our goal.

The challenges of this task are deeply linked with data sparsity and scalability, due to the often high dimensionality of the data, the imbalance, and the dependencies between labels that must be taken into consideration [10].

2.1.2 Text Classification

Text classification corresponds to the task of assigning the correct label or labels to a certain text sample. Examples of this task include language identification [11], genre classification [12], sentiment analysis [13, 14] and Spam detection [15]. In the problem described by this dissertation, the aim is to find

the correct labels to be assigned to each text document in the dataset.

Working with textual documents requires converting our texts to trainable data. This process is called vectorization and there are multiple ways of achieving it. Two common approaches to representing documents in vector form are bag-of-words (n-grams) and word vectors, and will be further explained in Chapter 3.

2.2 Portuguese Navy's Structure

The Corpus we are working on was extracted from documents provided by the Information Technology Superintendence (STI - *Superintendência das Tecnologias da Informação*), one of the functional units of the Portuguese Navy. The distribution of such correspondence concerns multiple units: STI, CDIACM, DAGI and DITIC. Besides these departments, it is also regarded in the distribution table the C/GAB unit, SERV.PART and ADJ.SEC1, corresponding to different correspondence participants within the STI. The Information Technology Superintendence is part of four directing units that work on resource management and are supervised by the Chief of Staff of the Navy.

The provided documents belong to three distinct groups, corresponding to the folders in which they are organized: Incoming (E), Internal (I) and Outgoing (S) depending on the sender and addressee. The Corpus has a total of 7300 documents with approximately 65% of such documents being in a standardized structure. The remaining documents are non-standard and correspond to invoices, receipts, faxes, memos, diplomas and emails [16].

2.3 Related Work

Working on the problem of Multi-label Classification, it is common to divide the approaches in two categories: Problem Transformation and Algorithm Adaptation. The former transforms the problem so that it can be solved by traditional classification algorithms while the latter makes use of such algorithms which are adapted to do Multi-label Learning. The main reasons for choosing Problem Transformation is the ease to test and to classify using many common used algorithms. Otherwise, we might be more interested in using Algorithm Adaptation approaches, considering these are often more suited to the problem and take into consideration things like label correlation, which tends to be overlooked in some Problem Transformation approaches [10].

Apart from Problem Transformation and Algorithm Adaptation, some authors [17] still hold a special category for solutions that make use of ensemble methods to predict labels.

2.3.1 Problem Transformation

In the category of Problem Transformation, some solutions while being conceptually simple to implement still reveal promising results. One example of such approaches is Binary Relevance [18, 19] which simply consists in training one binary classifier for each label. The advantages are the conceptual

simplicity, not being constrained to a particular learning technique, so almost every single-label classifier can be used as the underlying model, with models based in Support Vector Machines [20, 21] and Naive Bayes [21], being successful. Besides that, they are able to learn from partially labeled instances, since each classifier is trained independently. The major drawback is the scalability, considering the number of classifiers grows linearly with the number of labels, being unusable in very large datasets (Extreme Multi-label Classification) [18]. Another issue appointed to Binary Relevance is not taking into consideration label correlations, which are important for Multi-label classification [18].

There are some algorithms that base themselves in Binary Relevance and try to account for the correlation part. One of these methods is Classifier Chains [22] which similarly to Binary Relevance trains one classifier per label, however these classifiers are trained sequentially and take as input the instance to classify plus all the labels resultant of the previous classifiers. As mentioned, this has the advantage of accounting for the correlation while maintaining a computing complexity close to Binary Relevance, with the disadvantage of not being possible to do parallel training to reduce computing time, or train in incomplete data [22].

Other Problem Transformation methods include Label Powerset methods [19], which transform the Multi-label problem into a single-label one by training one classifier for each possible combination of labels. The advantage of this method is that it takes into consideration correlation between labels, but deeply suffers from the problem of data scarcity, since some label combinations might not have enough representation in the dataset and the results might not be distinguishable from Binary Relevance, despite being expected to show better results for taking into account label correlation [19].

For a performance comparison between the discussed methods, refer to Table 2.1, regarding the results obtained when doing Multi-Label classification in various datasets [23].

2.3.2 Algorithm Adaptation

For Algorithm Adaptation approaches, there are techniques altered in a way to directly solve the multi-label classification problem, and are generally better at taking things such as correlation between labels into consideration. Some Algorithm Adaptation may be bound to a particular learning method, however, they can be as simple as using a Problem Transformation method internally or collecting multiple classification confidences and joining them [22]. For such approaches we can see ones using traditional learning methods as well as recent deep learning approaches.

For classical classification methods, one can find adaptations for Decision Trees, Support Vector Machines and Instance Based Classifiers [10]. An example of the last method is an adaptation of k-Nearest Neighbors to the Multi-label classification [24] problem, a lazy learning method that shows very promising results in several datasets, as can be seen in Table 2.1.

When classifying a new instance, this method firstly finds the set of k nearest neighbours, by calculating the Euclidean distance between samples. k is a parameter for this model and according to the source [24] results are shown ranging from $k = 8$ to $k = 12$, which revealed that the number of nearest neighbours did not significantly affected the performance of Multi-label kNN (ML-kNN), having settled at

$k = 10$.

After selecting the closest neighbours, a counting its made for each label to reveal how many neighbours possess that label, and then resort to the maximum a posteriori (MAP) calculation to find the expected value for a specific label for our instance.

More recent methods using Artificial Neural Networks often focus on the scalability and data sparsity often associated with the Multi-Label learning problem, often referred to as Extreme Multi-Label Learning problems [9, 25]. Using such methods we see very different approaches, focusing on the loss functions resulting in a Precision@5[1] of 48.08% in the *EUR-Lex*[2] dataset [25], using different types of Artificial Neural Networks, such as applying word embeddings followed by a Convolution Neural Network resulting in a Precision@5 of 51.41% in the same dataset [9] or using restricted Boltzmann machines that help improve the feature-space [26], this one not focused at Extreme Multi-Label, resulting in an accuracy of 0.742 against 0.770 compared with ECC in the *Medical* Dataset, 48.0% to 45.4% on *Enron* Dataset and 45.1% to 46.1% on *Reuters*, all these text datasets [26].

2.3.3 Ensembles

Other examples that are often included in a group of its own and are known for increasing overall accuracy, overcoming over-fitting and allowing parallelism are Ensemble Techniques [22].

One such example of this method is the Ensemble of Pruned Sets [27]. For this method, we begin by building pruned sets, which consist of only the most relevant label relationships in the training set. After pruning the least frequent sets, we are left with the label groups that have a significant representation in the set. After that, a binary classifier is trained for each set, similar to what happens with Label Powerset. The additional advantage of including an Ensemble is reducing overfitting and the possibility of assigning sets of labels that were nonexistent in the training data [27].

The Ensemble of Classifier Chains [22], as the name points out, is a grouping of Classifier Chains, and in training, each chain is assigned a random chain ordering and a random subset of the dataset [22].

One more Ensemble method is RAkEL [28], or *RAndom k-labELsets*, which in its turn, is a grouping of Label Powerset classifiers. For this method, we have two changeable parameters, k and m, which represent the length of the label sets to be considered and the number of iterations, respectively.

From the results presented at Table 2.1 we can see that between the studied methods, the one that performed better (being the best method in 6 of the 7 sets) is Ensemble of Classifier Chains. This method models correlations efficiently and being an ensemble is not prone to over-fitting, performing well in various datasets [22]. Other methods, namely Ensemble of Pruned Sets and ML-kNN also came very close to the best in nearly every set, being both the best method in 2/7 datasets.

[1] metric mostly used in Information Retrieval problems, measures the ratio of relevant results in the top five predictions – in this case the five labels with higher confidence

[2] *EUR-Lex* Dataset with 19348 instances, 3993 labels and a label cardinality of 5.31 [25]

Method	Computers	Education	Entertainment	Health	Recreation	Reference	Medical
BR	0.054	0.060	0.082	0.052	0.087	0.040	0.011
LP	0.061	0.065	0.087	0.058	0.097	0.045	0.014
PS	0.058	0.061	0.085	0.056	0.092	0.043	0.013
CC	0.056	0.061	0.083	0.055	0.094	0.041	0.011
ML-kNN	**0.037**	**0.040**	0.057	0.042	0.057	0.029	0.016
EPS	**0.037**	0.042	0.055	0.038	0.058	**0.028**	0.012
ECC	**0.037**	0.041	**0.053**	**0.036**	**0.056**	**0.028**	**0.010**
RAkEL	0.041	0.043	0.064	0.040	0.061	0.029	0.011

Table 2.1: Hamming Loss for some of the discussed methods (less is better) with best results for each dataset in bold font. Extracted from *Hybrid Noise-Oriented Multilabel Learning* [23]

Chapter 3

A Classifier System for Correspondence Distribution

In order to build a reliable classification system using machine learning there are several steps that must be followed in order to obtain valid and reliable results. The focus of this chapter is to enumerate such steps and to describe how they were performed in order to reach the proposed solution for an automatic correspondence distribution method tailored for the Portuguese Navy.

In this chapter, we will discuss the steps that must be traversed in order to implement and evaluate a classification model, in specific a Multi-label classification method, where each label has multiple degrees.

For that reason, this chapter is split in four sections that detail the engineering part of building a classifier. In Section 3.1, we will discuss the data provided which was used to train our models. In Section 3.2, we will talk about important visualization methods and feature engineering tools. In Section 3.3, an analysis on each of the models built for the problem will be detailed, along with motives for their choice.

3.1 Data Collection

Whichever machine learning model we are training, the first and most important item that we must consider is data. Without data no model can be trained, since we have no observations to base our predictions off and so, no methods to extrapolate and make predictions for new observations.

In our case, the data comes from PDF documents belonging to the Portuguese Navy. These documents were processed into textual data which was then used for training the models studied by this work. This conversion to text was studied and performed as part of another dissertation [16].

This collected data is used for training, and can either be labeled or unlabeled. In our case, the data is labeled since each document in the dataset possesses a table with our target: the action degree required for each of the possible recipients of the document.

The provided documents correspond to the correspondence sent and received by the STI and its

sub units. The distribution in time encompasses documents from 2014 up to 2019 [16]. This periodical distribution was subsequently used to evaluate our best performing model, saving the unseen documents of 2018 and 2019 for evaluation purposes. In Table 3.1 we can see the distribution of the number of documents, according to their year. Setting aside 2018 and 2019 leaves us with a test set with around 26% of the documents from the dataset.

Table 3.1: Distribution of documents by year.

Year	# Documents	Ratio
2014	446	6.11%
2015	1655	22.67%
2016	1778	24.36%
2017	1498	20.52%
2018	1415	19.38%
2019	508	6.96%

3.2 Data Preparation

The dataset used through the described tests was created from PDF documents made available by the Portuguese Navy, which were then converted to text using Optical Character Recognition. After the PDF files were converted into text files irrelevant data was removed and the text fields were segmented to be grouped into distinct tags in a XML file. The existing tags are the following:

- Header
- Distribution Table
- Process Number
- Title
- Body
- Document Number
- Reference
- Recipient
- Signature
- Attachment

Looking into each tag, we can select and discard a few. Distribution Table and Process Number are target values and for that reason should not be included in our training data. Document Number is a document identifier that should not be considered either.

It is also important to stress that the listed XML tags are only being parsed in the present dataset for normalized Navy's documents. Layouts that deviate from the standard of the organization are not

recognized and so all the content recognized by the OCR is put to a single XML tag containing the whole body of text.

Classification using multiple tag combination hypotheses was conducted, in order to find the best set of features to be used by our classifier, one that could predict accurately the correct classification, with as little redundant data as possible.

Through the elaboration of this dissertation, the Corpus was gradually improving, with samples added and noise removed frequently. The last available version of the Corpus is comprised of 7300 documents, grouped by year and belonging superintendency (*STI*, *CDIACM*, *DAGI* and *DITIC*), and identified by a unique incremental identifier.

Regarding the task of predicting the distribution of each document, the target was present in each document in the Table of Distribution field, as a list of 7 characters, identifying each of the departments through which documents are circulated within the Navy. These departments correspond to the target labels and are the following, in Portuguese (meaning between parenthesis):

- STI - Superintendência das Tecnologias da Informação (IT Superintendency)
- CDIACM - Centro de Documentação de Informação e Arquivo Central da Marinha (Navy's Central Information and Archive Documentation Center)
- DAGI - Direção Análise e Gestão de Informação (Direction of Analysis and Information Management)
- DITIC - Direção de Tecnologias de Informação e Comunicações (Information and Communication Technologies Management)
- C/GAB - Gabinete do Superintendente das TI (IT Superintendent's Office)
- SERV.PART - Serviço Particular (Private Service)
- ADJ.SEC1 - Entidade Contabilística (Accounting Entity)

Other departments are also sporadically displayed in the distribution table, however not enough times to be considered relevant and so their presence was discarded:

- DAP - Departamento de Apoio (Support Department)
- PMO - Escritório de Gestão de Projetos (Project Management Office)
- DSUP - Depósito de Suprimentos (Supply Depot)

After having enough data to train, it is important to make visualizations and calculate descriptive metrics for our dataset. This helps to get an idea of the kind of data we are dealing with, in order to look out for biases and outliers, and to prevent issues that might hinder the success of our classifiers.

For example in this particular task, there were residual departments that only were referenced a couple of times in the distribution table in the whole spectrum of documents, not possessing enough relevance to be included in our training model, being for that motif discarded.

Table 3.2: Group of key metrics from the Portuguese Navy dataset

Metric	Value
Number of Samples	7185
Number of Labels	7
Number of Classes per Label	3
Average Samples **blank**	4995
Average Samples **C**	1892
Average Samples **A**	298
Median Number of Words per Sample	50

Also during this step, action degrees such as *A1*, *A2* and *C1*, *C2* were converted into *A* and *C* respectively, as they were used in the past by the organization and afterward deprecated.

Some metrics related to our dataset which are considered of value, are described in Table 3.2. These metrics represent the dataset after removing the residual labels mentioned and converting the action degrees. From the initial 7300 documents, 115 do not have a distribution table, so are discarded. In this table an imbalance between classes is noticeable in our samples, since labels *blank* and *C*, with an average 4995 and 1892 samples respectively, have much higher representation than label *A* with only 298 average samples per label. The median number of words per sample in the Table refers to the XML fields selected for our experiments, being composed by Title and Body of standardized documents and Text field on non-standard formats.

This uneven distribution can be further and better understood by visualizing the following Figure 3.1, which represents the distribution of classes within each label. From this graph, we can see that label *DITIC* is the only one that deviates from a majority class *blank* and minority class *A*. For this label *C* is the most represented class and classes *blank* and *A* are relatively close, even though class *A* is the least represented in every label, with labels *CDIACM* and *ADJ.SEC1* counting only with twelve and four class *A* samples, respectively.

During this phase, specially in problems related to text classification, it can also prove useful to study the vocabulary used and the length of samples. In Figure 3.2 we can visualize the hundred most common words found in the documents, after removing stop-words and lowering the case. By analyzing these words we can notice the most used vocabulary is quite specific to the context of the Navy and managing national and international affairs.

A representation of document length by word count is presented in Figure 3.3. In this histogram, we can see that while the majority of documents have under 1000 words there are several documents with much higher length, deviating from the standard values. The median document length for our dataset is of 50 words, considering Title and Body of the sample, or 170 if we consider every document tag. These metrics and the document statistics are the result after removing Portuguese and English stop-words from the text documents.

3.2.1 Document Representation

Another important consideration, since we are working with text documents is how to represent our files in a way that can be interpreted by our models. This step, called vectorization has the purpose of

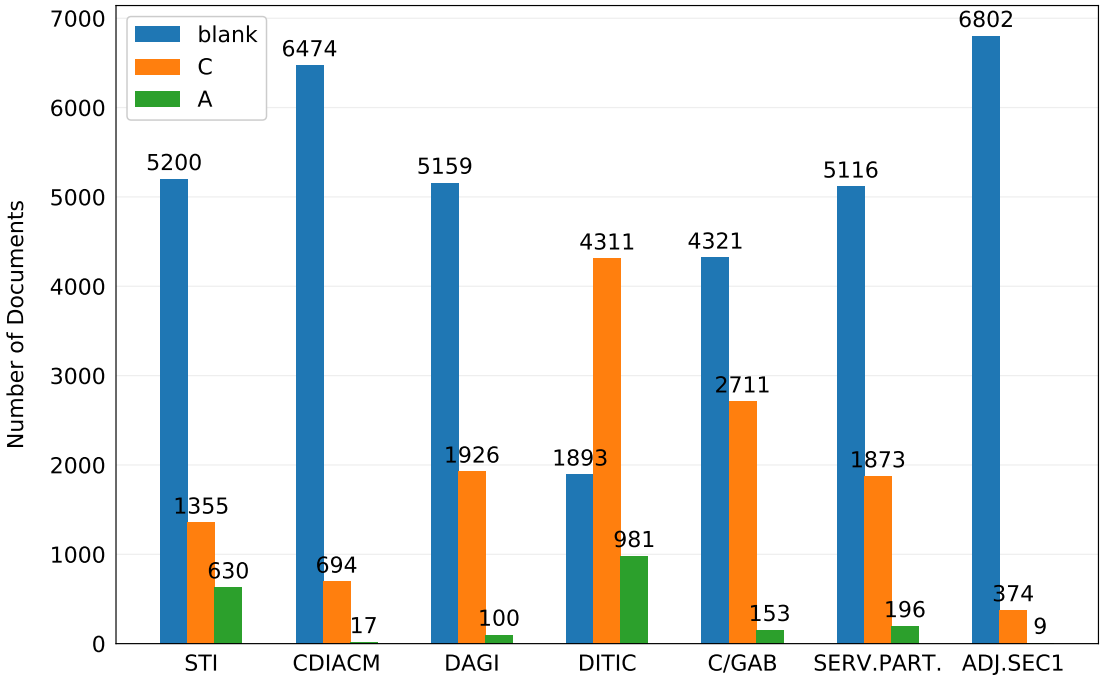

Figure 3.1: Bar plot with the distribution of classes for each department in the dataset.

converting the textual data into numerical vectors that can be subsequently used in training. How we choose to represent our samples affects every subsequent step of the classification.

When working with text classification models, one common approach is to use Bag of Words techniques, also known as N-gram models to convert the documents into vectors. This works by assigning each token a position in the vector and filling that position with the count for the number of times the token appears in the document. In the end, we have for each document a vector containing the number of times each known token appears in that specific document. Using this model we ignore word context and sequence in the document, reducing computing time at the expense of some information loss.

Alternatively, instead of having a vector with frequencies, it is also common to attribute values to each term in the vocabulary. For example, when using tf-idf we calculate the weight of each term which is then using instead of flat frequencies. The tf-idf algorithm is a product between term frequency(tf) – the frequency a word t appears in a document d – and the inverse document frequency (idf) – the fraction of documents in which the word t appears [29]. This is a better alternative to using a count vectorizer being used through this work's experiments.

The disadvantage of N-gram models is not taking into account order in the sequence, which results in information loss. To prevent this loss in text classification it is common to use sequential models. For these models, each token is represented as a vector and the document to represent is nothing more than a sequence of vectors.

How the tokens are vectorized presents the big distinction for these models. On the one hand, we have the classic approach using One-Hot Encoding - attributing one position in the vector for each token in the vocabulary and filling it with a value corresponding to that token's occurrence.

On the other hand, we can make use of Word Embeddings. This method, which recently has been

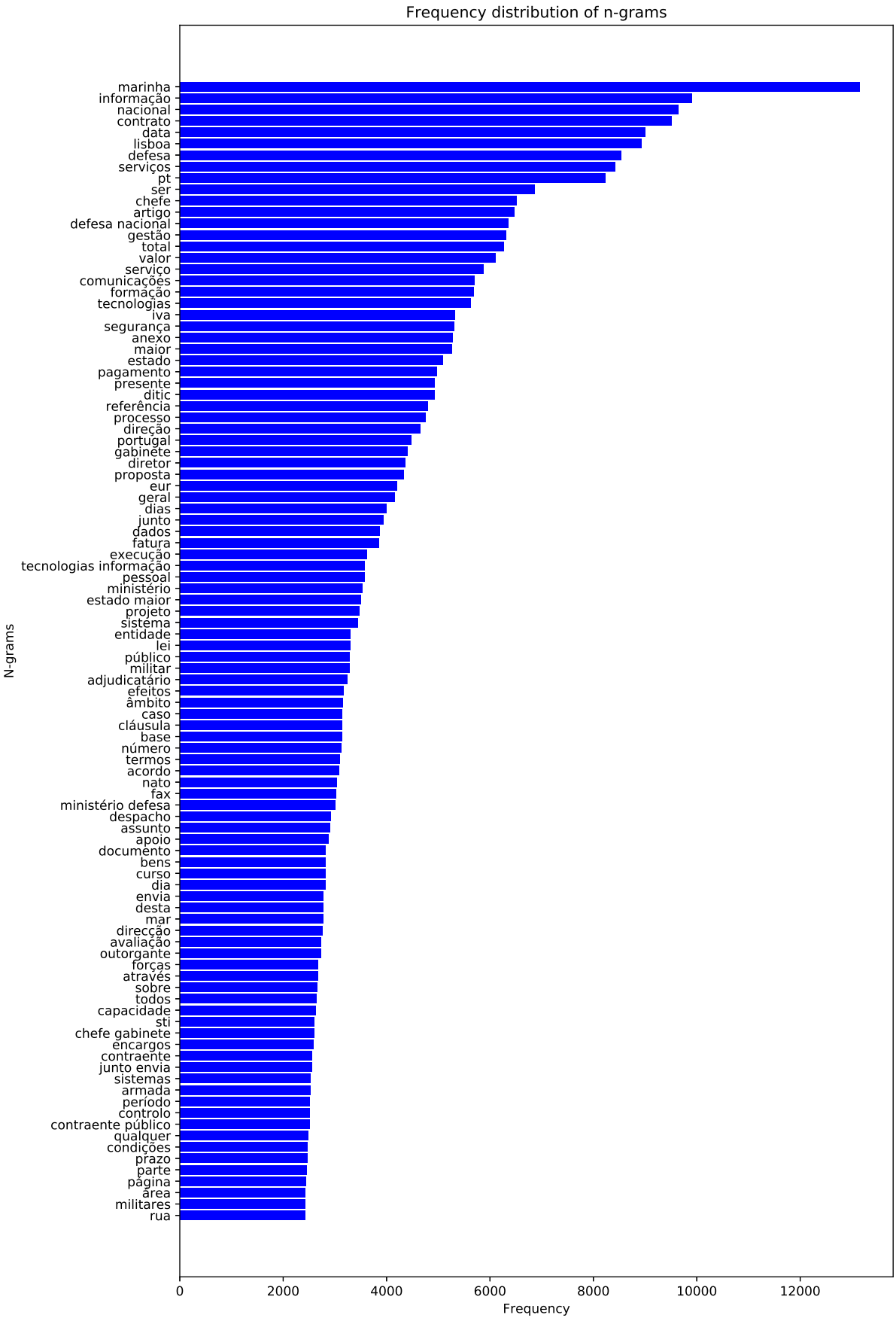

Figure 3.2: Representation of the distribution of the 100 most common uni and bi-grams within the dataset (excluding stop-words).

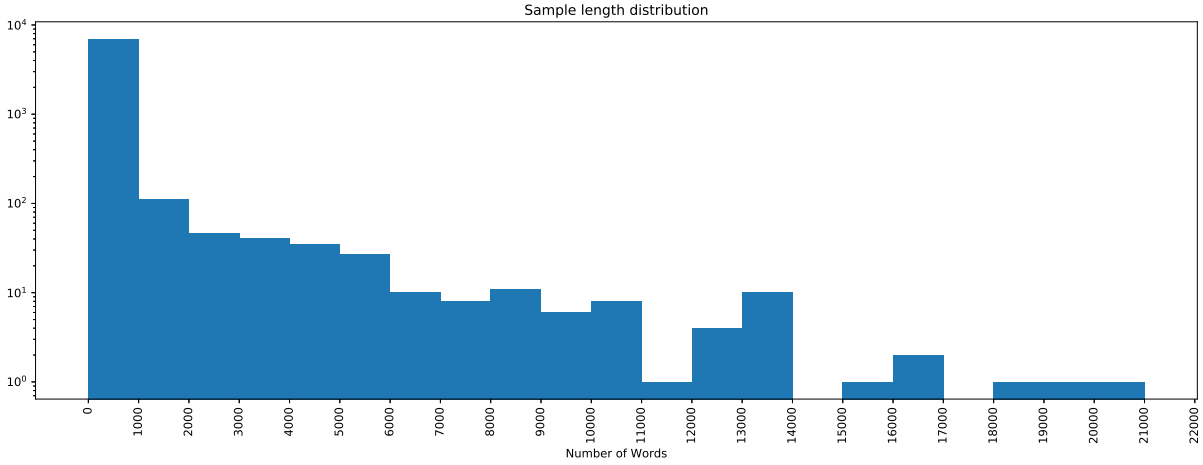

Figure 3.3: Box plot for the length distribution of the documents in number of words. Considering only Title, Body and Text tags after having stopwords removed. Frequency scale is logarithmic.

widely used consists in giving each word a dense vector representation, representative of its position in semantic space. Such embedding weights can be trained in the dataset from scratch, be imported from another Corpus or be pre-trained in one Corpus and then fine-tuned for the dataset in question through transfer learning.

3.3 Model Selection

Looking into the approaches discussed at Section 2.3 and transferring them to our challenge, two distinct ways to tackle the problem rise.

The first suggested approach consists in splitting each individual department degree (Blank, A and C) into separate labels, while accounting for the exception that each department can only have one degree and then treating it just like a Multi-label Classification problem.

The other approach, and the one more exhaustively pursued throughout this dissertation was to treat the classification as in the Binary Relevance method described in Section 2.3.1, considering each label an independent multi-class classification problem, selecting for each department one degree, either *A*, *C* or *blank*.

After some analysis over the dataset, the major obstacle to a robust classifier, as can be seen in Table 3.2 and Figure 3.1, is the imbalance in data, given the lack of samples for class *A* for some departments.

After watching such good results using a bag-of-words approach, the decision to try sequential models was made, given their recent success in the field [1, 2]. For this reason we tested with embeddings and a Separable Convolutional Neural Network (SepCNN). The experiments made included embeddings trained from scratch, and using FastText's pre-trained word embeddings for Portuguese [30, 31], trained on Wikipedia data[1]. FastText embeddings were both used out-of-the-box and fine-tuned to fit our dataset.

[1] https://dumps.wikimedia.org

In order to achieve the best possible results, the following models were analyzed:

3.3.1 Baseline Classifier - Most Frequent Tag

In order to better assess the performance of the implemented models, it is a common practice to develop some kind of baseline classification method. For this project, our baseline approach consisted of a dummy estimator that assigned the most common tag in training to every document to predict. Though this classifier is very rudimentary, it is a good benchmark to evaluate other models.

3.3.2 Logistic Regression

Logistic Regression is a classification method that fits the data through a linear regression model and then computes the probability of our sample belonging to each class. It expands the concept of linear regression in order to solve the binary classification problem.

Logistic Regression outputs a value between 0 and 1, corresponding to the probability of a sample *x* belonging to a particular class. This is achieved by resorting to the sigmoid function [32]:

$$f(x) = \frac{1}{1 + e^{-x}}$$

In order to fit our data samples, the Logistic Regression model is the following [32]:

$$f_{w,b}(x) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-(wx+b)}}$$

This equation gives us the probability of a sample *x* belonging to a class in a binary classification. All that's left to do is to find the optimal parameters w* and b*. Such variables can be found by computing the Maximum Likelihood. The likelihood is computed via the formula [32]:

$$L_{w,b} \stackrel{\text{def}}{=} \prod_{n=1}^{N} f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)}$$

In practice, since the exponential is being used, it is more convenient to maximize the log-likelihood instead, which in turn is defined by [32]:

$$LogL_{w,b} \stackrel{\text{def}}{=} \ln(L_{w,b}) = \sum_{n=1}^{N} y_i \ln(f_{w,b}(x_i)) + (1 - y_i) \ln(1 - f_{w,b}(x_i))$$

Logistic Regression has displayed great performance in text classification tasks, especially when compared with other classic methods [33].

3.3.3 Naive Bayes

Naive Bayes is based on the assumption of independence between features and applying Bayes' Theorem. It calculates the probability of a given sample belonging to each particular class in a simplistic

way, by taking into consideration the prior probabilities for each class and the conditional probabilities of seeing the input, given each class.

In Naive Bayes, we construct a belief network based on prior observations x – our samples – and the corresponding class c, according to the formula [34]:

$$p(x, c) = p(c) \prod_{i=1}^{D} p(x_i|c)$$

After this step we can use Bayes' Theorem to compute the probability of a new sample x' belonging to each of the possible classes with [34]:

$$p(c|x') = \frac{p(x'|c)p(c)}{p(x')} = \frac{p(x'|c)p(c)}{\sum_c p(x'|c)p(c)}$$

Despite its simplicity, Naive Bayes can be used for text classification and provide good results. One such example where Naive Bayes is often used is spam filtering [35]. This classification task results better when particular terms found in the vocabulary are often associated with a particular class.

3.3.4 Support Vector Machines

Support Vector Machines (SVMs) work by finding the boundary that better discriminates classes. This boundary is a linear hyper-plane. However using kernels this method can distinguish between seemingly non-separable data by setting it in higher-dimensional space, where data might be more easily separated. SVMs attempt to minimize the generalization loss by choosing a separator that is the farthest from each of the classes [36].

Typically, this classifier works with two classes in a Binary Classification problem. Since we have three classes the problem is solved by training one classifier for each class in a One Versus All manner.

3.3.5 Decision Trees and Random Forests

Tree based models work by partitioning data into cuboid regions. Classification is done by traversing the binary tree sequentially. Issues recognized are the tendency to overfit a problem, especially in deep trees. For that reason the following model is often known to provide better results.

Random Forests is an ensemble of Binary Decision Trees, each trained in a subset of our dataset. The results of the classification from each tree are then counted and the majority vote is selected as the classification result.

3.3.6 k-Nearest Neighbors

Despite being conceptually simple, this method often provides very high performances. It can be used as long as there is a distance metric to compare between each sample of the dataset.

Its training cost is nonexistent, since all computations are made during the classification of each new instance, which often results in a higher classification cost. For every new sample, this algorithm

calculates the distance to each other sample on the dataset, returning the closest *k* nearest neighbors (hence the name) after which it classifies the new sample as an average between its neighbors. Often this average is weighted in function to the distance from the sample to classify to its neighbors, so that closer samples have a higher vote in the classification.

3.3.7 Multi-Layer Perceptron

A Multi-Layer Perceptron is an instance of an Artificial Neural Network, composed of multiple layers of Perceptrons that work in a Feedforward Network.

For the experiments, *Tensorflow* Library was used, in order to build a Multi-Layer Perceptron with two Dense Layers with a Dropout rate of 20% and 64 neurons in the hidden layer.

3.3.8 SepCNN

SepCNN is a neural network specialized for text classification tasks that works with word embeddings and uses one-dimensional depthwise separable 1D convolutional layers. Depthwise separable convolutions reduce the computation time needed for the convolutions and the number of parameters to tune. They work by dividing the traditional convolution into a depthwise convolution (filtering step) followed by a pointwise convolution (combination step). Such layers have been used in some recent architectures and show great promise.

Chapter 4

Results

The present chapter aims to display the results for each of the experiments carried out during the course of this dissertation. Initially, in Section 4.1, a description of how the tests were performed and evaluated will be provided, followed by some considerations about the outcome of the experiments conducted during the work done through this dissertation, in Section 4.2.

After that, an analysis of the emerging challenges of this task are described, in Section 4.3, and after that a solution proposal is presented and tested, in Section 4.4. Finally, the results are consolidated after being once more evaluated, in Section 4.5, and a summary of the whole Chapter is presented in Section 4.6.

4.1 Methodology and Evaluation

All models were compared using the same methodology. Initially a random split was made for a train and test static sets that was the same for each of the experimented models, for convenience and in order to have a reference performance for further evaluations.

Subsequently, a test set was detached from the remaining documents which contained the most recent documents of the dataset (those belonging to the year 2018 and 2019). The remaining elements of the dataset were divided into 4 splits to be used for cross validation.

Several evaluation metrics were used in order to better evaluate the results. The most referred to were accuracy, exact match ratio and recall.

Given a Multi-label classifier h, let Y_i be the ground truth set of labels we aim to predict and Z_i the set of predictions made by our classifier h

- Exact Match Ratio (EMR)- Measures how many instances were completely well classified. Ignores partially correct instances [37].

$$\mathrm{EMR} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

- Accuracy - Measures the proportion of correct labels [37].

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

- Precision - Measures the fraction of the predicted labels that were actually correct [37].

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|}$$

- Recall - Ratio between the predicted correct labels and all true labels, averaged over all instances. A low Recall value in minority classes might reveal a bias towards the majority classes [37].

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{Yi}$$

- F1-Score - Calculates the harmonic mean between precision and recall [38].

$$\text{F1} = \frac{1}{n}\sum_{i=1}^{n}\frac{2\,|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

- Hamming Loss - Loss function that accounts for prediction errors (false positives) and missing errors (false negatives) [38], computing the percentage of labels whose relevance is predicted incorrectly [38, 39].

$$\text{Hamming-Loss} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \triangle Z_i|}{|n|}$$

4.2 Discussion

In this Section, some considerations about the outcome of the experiments conducted during the work done through this dissertation will be presented.

As discussed, in order to solve our classification problem, multiple classifiers were implemented and tested, namely Logistic Regression, Naive Bayes, Support Vector Machines, Decision Trees, Random Forests, k-Nearest Neighbors, Multi-Layer Perceptron and Sequential SepCNN using weights trained from scratch, loaded from FastText and out of the box and tuned from FastText weights.

The cross-validated results are described in Table 4.1. The results for each classifier along with a description of each design choices can be read below.

Table 4.1: Results for the classifiers trained in the Portuguese Navy dataset (cross-validated).

	Baseline	NB	LogReg	MLP	SVM	Rforests	Decision Tree	kNN	SepCNN	Sep FT	Sep FT Train
EMR	15.56%	22.09%	**49.23%**	46.86%	48.00%	46.73%	36.95%	37.43%	30.85%	23.26%	32.68%
Accuracy	74.52%	80.16%	**90.08%**	89.41%	89.59%	89.22%	86.40%	86.20%	84.50%	80.61%	85.07%

4.2.1 Baseline Classifier - Most Frequent Tag

The baseline classifier was the worst performing classifier between the implemented models. Despite this fact, it is quite interesting to see how a simple classifier, ignorant of any features displays nearly 75% accuracy over all labels. This is a display of the imbalance in data. Despite this fact, it only fully correctly predicts little more that 15% of the documents.

4.2.2 Naive Bayes

Naive Bayes is often used as a baseline method, given its simplicity in implementation. It is not a surprise to see its performance in the lower spectrum of the competing classifiers given the fact of not taking into consideration dependencies between features/words. Its classifications were bottom three for every one of the seven labels to classify, giving him an average accuracy of 80.16% and the lowest exact match ratio within the group of classifiers without taking the Baseline model into consideration with only 22.09% of the samples being correctly predicted for all labels.

4.2.3 Logistic Regression

As visible in Table 4.1 this was the classification model that achieved the best overall results. With over 90% average label accuracy, it was the best classifier for two out of the seven labels and came in top three for every label. It also displayed the best Exact Match Ratio, predicting 49.23% of the documents completely correctly through all 7 labels.

4.2.4 Multi-Layer Perceptron

Multi-Layer Perceptron was among the top three classifiers for this problem. With 46.86% Exact Match Ratio and 89.41% Average Accuracy, it provided results very close to both Logistic Regression and Support Vector Machines, as is visible in Table 4.1.

4.2.5 Support Vector Machine

The Support Vector Machine classifier was able to attain good results, displaying the best accuracy for the label *DAGI* and showing the third best average accuracy of 89.59% (closely after Logistic Regression and Multi-Layer Perceptron classifiers) and the second best Exact Match Ratio (48.00%).

4.2.6 Decision Trees and Random Forests

Decision Trees were on the lower side of performance as well. This is, however, not surprising given the problems appointed before, such as the classifier's tendency to over-fit, which cost it some accuracy.

Random Forests on the other hand overcame this problem and performed well, with 89.22% average accuracy and 46.73% Exact Match Ratio.

4.2.7 k-Nearest Neighbors

While working with kNN, some parameters were tested in order to optimize this model for our problem. For this model, such parameters involve majorly the distance metric to be used (in our case we opted for the commonly used Euclidean Distance) and the number of neighbors that account for the classification and whether to do a weighted average or not. Since the answer to this questions is not always obvious, it is common to plot a Elbow graph in function to the number of neighbors, in order to find the sweet spot.

As it can be seen in Figure 4.1, we can see distance based weights (in blue) outperform uniform weights (in red) and we see that for more than three neighbors, accuracy increases a little and stabilizes, before dropping, whereas Exact Match Ratio tops at three neighbors, meaning that the increase in accuracy causes less classification errors, but more documents with errors, which might indicate overfitting. The difference between uniform and distance-based weights on its turn can be justified by the sparsity between samples in the dataset. Note that these results are from the initial random split, and not the cross-validated scores present in Table 4.1.

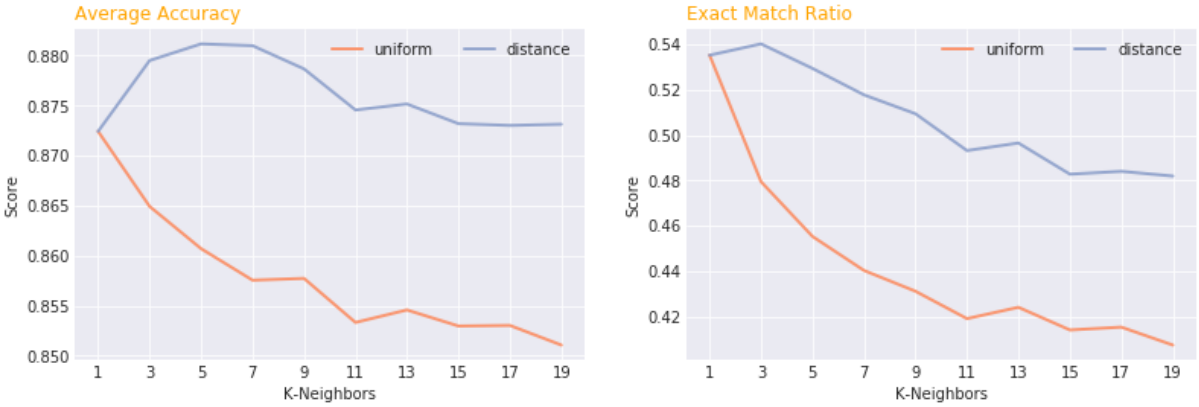

Figure 4.1: Elbow graph for the K-Nearest Neighbors algorithm

Looking into k-Nearest Neighbors performance during cross-validation, we can see it was not between the best performing classifiers, with 37.43% Exact Match Ratio and 86.20% Average Accuracy.

4.2.8 Sequential Networks

All sequential networks trained for this dissertation used the architecture of SepCNN. Three sequential models were trained and its performances evaluated following the same procedures. Initially we trained the word vector weights from scratch (SepCNN), then we used FastText weights out of the box (Sep FT) and finally, FastText weights were tuned in our dataset, by freezing the network's learning while adjusting the weights (Sep FT Train).

From looking into the results present in Table 4.1 we can see SepCNN with FastText vectors out of the box was the worst of the three classifiers. This shows that FastText vectors, despite being trained in a much bigger Corpus were still beaten by word vectors fitted strictly in the training set. However, when pre-training the same FastText word vectors the results exceeded training from scratch, showing an advantage in training in a larger Corpus, with an average accuracy increase of approximately 4.4%

over FastText raw vectors and 0.5% average accuracy increase over trained from scratch word vectors.

When working with sequential models, using word vectors led to results below average, despite the rising popularity of this technique. Reasons for this seem to be linked to the specificity of the vocabulary, being most of it related to Navy's themes and the objective concise writing removes most of the ambiguities that would make word vectors trained in a large Corpus overcome other methods. Word vectors seem to work best when presented with a wider vocabulary and ambiguous texts, where a context is important to extract a meaning.

Another factor is the ratio between the number of samples and number of words per sample, which according to this source should be taken into consideration. When this ratio is higher than 1500, SepCNN seems to outperform Multi-Layer Perceptron and other classic methods. Below that value, Multi-Layer Perceptron performs better [40]. Taking our dataset statistics, presented in Table 3.2, into consideration, this ratio is of approximately 145, roughly 10 times less than the suggested value, indicating a shortage of samples to train an appropriate sequential model.

4.3 Error Distribution

In order to get some insight as what type of errors were being made by the experimental models, and as a means to improve the results, we plotted the confusion matrix present in Figure 4.2. This Figure represents the results for the best performing classifier which was the Logistic Regression model.

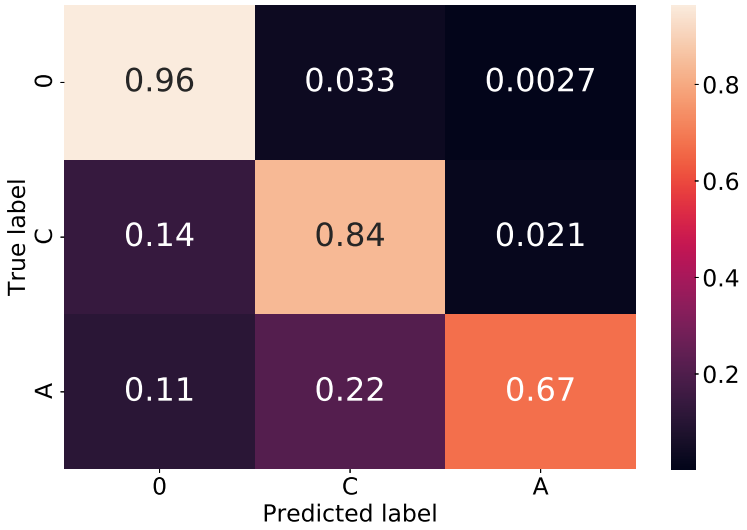

Figure 4.2: Normalized Confusion Matrix for the best performing classifier averaged over every label.

From Figure 4.2 we can see some results that were to be expected: majority classes blank (0) and C are the most accurately predicted, as it is noticeable from the high values in the diagonal of the matrix. We can also see that only 65% of our samples with degree A are being correctly classified, with 24% of the samples being wrongly classified as degree C and 11% as degree A. This is attributed to the

imbalance of the data. In Figure 4.3 the confusion of each label's classifier can be better seen and it is noticeable a clear bias in SERV.PART. label, given the high ratio of labels wrongly predicted as blank.

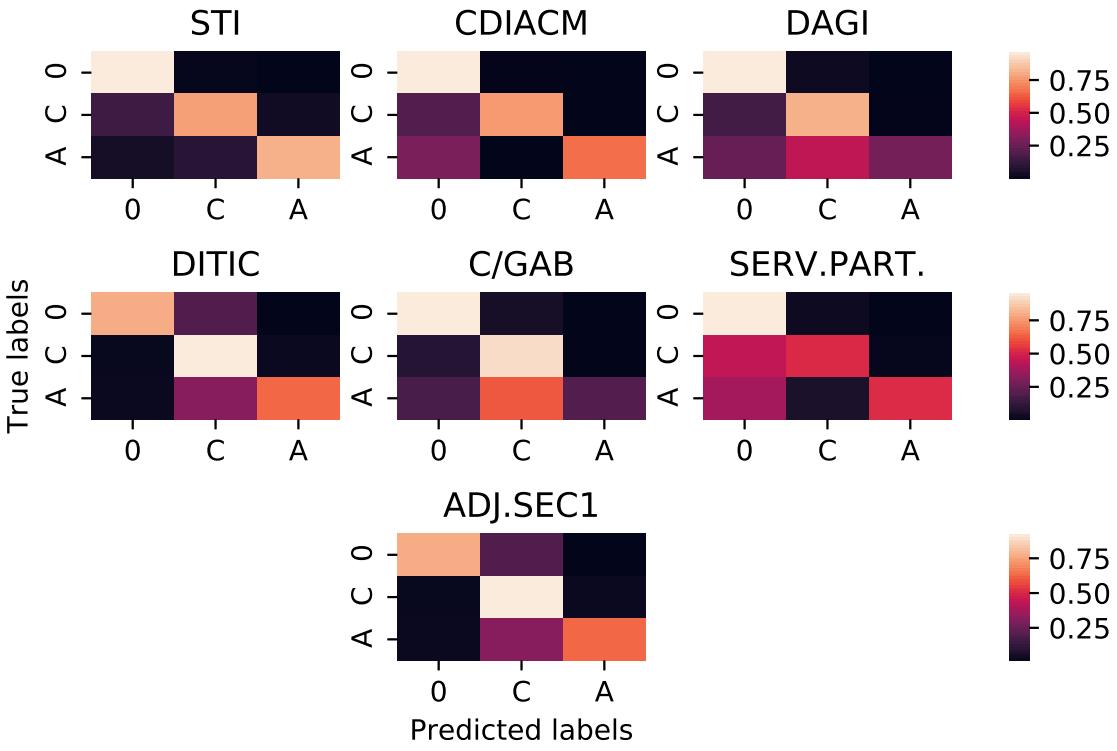

Figure 4.3: Normalized Confusion Matrix for the best performing classifier for each label.

4.4 Dataset Imbalance

As mentioned previously, one of the greatest obstacles when working with this dataset is the imbalance between labels. In Figure 3.1 we can see that for most departments action degree **A** representation is minimal, and this issue is even more prominent for departments *CDIACM* and *ADJ.SEC1*.

For this reason, it was important to study more in depth, methods to fight such imbalance. With this in mind, the top performing model was selected (Logistic Regression) and tested with a selection of data balancing methods. The results can be seen in Table 4.2.

4.4.1 Imbalanced Set

Doing imbalanced training means no alterations were made to the original set distribution. Training was made using the original distribution that is specified in Figure 3.1. This method serves as baseline for comparing the results obtained by using dataset balancing methods.

This method had the best overall accuracy and exact match ratio as seen in Table 4.2, which is expected, since other methods work by increasing the bias towards the minority classes, and that may

lead to failing more samples from the majority class, which since being more represented has a bigger impact in the accuracy score.

4.4.2 Undersampling

Undersampling aims at balancing sets by ignoring certain samples. This way, no fake data is introduced and as such no artificial biases are added to the set. However, a major disadvantage is the fact that we are discarding potentially useful information, from the samples that are being discarded. For this reason it should not be used in datasets with classes with very small representation (such as the study case), since that leads to heavy information loss.

For this experiment, the Undersampling technique in use was to set the number of samples equal to the minority class. This meaning that for example in *ADJ.SEC1* the number of samples was reduced from 4873 (4610 *blank*, 259 *C* and 4 *A*) to a mere 12 samples (4 for each class, matching the minority class *A* with 4 samples) discarding approximately 99.75% of the existing samples. This explains the poor results from this method for *ADJ.SEC1* as seen on Table 4.2 with only 19.58% accuracy for that label.

4.4.3 Oversampling

Oversampling is considered to be the counterpart of Undersampling. This balancing method works by resampling the minority classes. The experimental method used resamples the minority classes until the number of samples for these match the number of samples from the majority class. Using the same example from before (label *ADJ.SEC1*, the minority class *A* will be resampled until it reaches 4610 samples meaning these samples will weigh approximately 1152 times more than they originally did. This can raise problems such as overfitting to the minority samples and a higher susceptibility to outliers in such classes. Despite this, we can see in the experiments the results for oversampling were very close to the imbalanced set (less than .5% in average accuracy and over 1% accuracy in ADJ.SEC1 label's case.

From this, it can be concluded that the resampled instances were good representatives of the class and did not reduce accuracy heavily, despite the bias placed towards the minority classes.

4.4.4 EasyEnsemble

EasyEnsemble [41] aims at removing the information loss from Undersampling. It works by creating multiple subsets from the original dataset where normally the minority class is reused in each subset and the majority classes undersampled, in order to train one classifier per subset and subsequently classifying the new instances via a combined vote.

4.4.5 Data Balancing Results

In table 4.2 we can see the results for each one of the data balancing methods.

Table 4.2: Results for Data Balancing methods from a random sample.

	Logistic Regression			
	Imbalanced Set	UnderSample	OverSample	EasyEnsemble
STI	90.71%	86.55%	**90.92%**	87.76%
CDIACM	**96.67%**	70.35%	96.04%	76.97%
DAGI	91.09%	72.72%	**91.21%**	74.89%
DITIC	**85.38%**	78.76%	84.34%	80.72%
CGAB	**89.55%**	75.43%	89.13%	80.34%
SERV PART	**85.71%**	66.56%	85.21%	75.93%
ADJ SEC	**95.96%**	19.58%	94.79%	45.86%
Average Accuracy	**90.72%**	67.13%	90.24%	74.64%
EMR	**59.64%**	4.37%	58.31%	21.57%

As can be witnessed in Table 4.2 none of the methods performed better than the imbalanced set, however Oversampling provided really close results. This means it can be considered as a viable option to creating a balanced training in an imbalanced set so that every class has the same weight in the decision. This fact is more evident when looking into the confusion matrices present in Figure 4.4, as an average between labels and Figure 4.5, representing each label individually. It is visible that Oversampling is a good balancing method if the purpose is to increase recall and the number of positive classifications for the minority classes, at the expense of some accuracy.

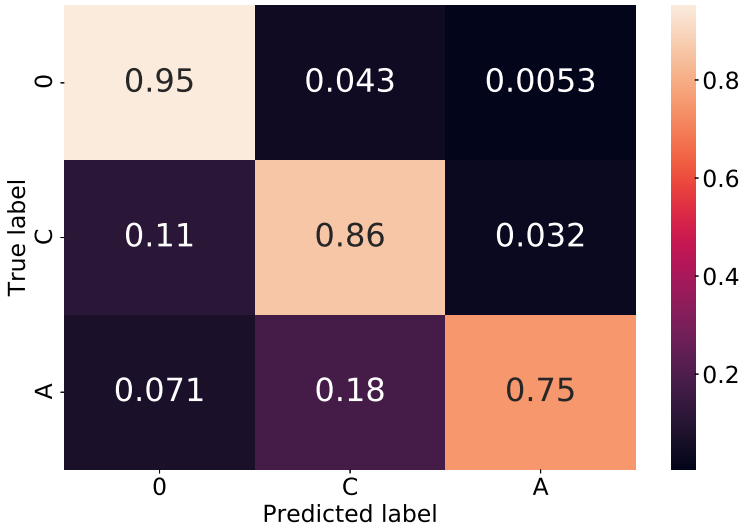

Figure 4.4: Normalized Confusion Matrix for the best performing classifier averaged over every label on an oversampled set.

Another thing to notice is the considerably lower performance of the Undersampling approach, reason for this is the residual number of samples for each class in some cases and the huge information loss resulting from not taking into consideration every available sample.

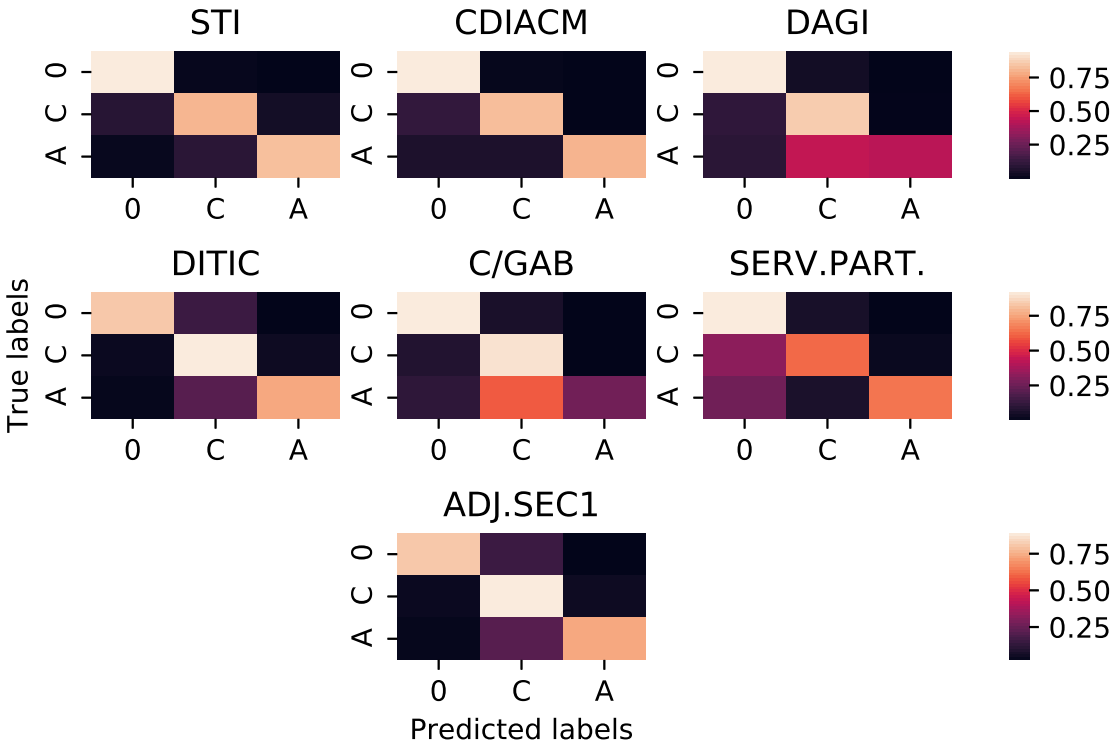

Figure 4.5: Normalized Confusion Matrix for the best performing classifier for each label on an oversampled set.

4.5 Final Evaluation

Since we wanted to avoid creating a bias towards any of the classifiers, we isolated a test set, comprised of the samples from the years 2018 and 2019. Training then our top classifier in the whole training set, using the same parameters as in development we obtained an average accuracy of 82.02% and an Exact Match Ratio score of 35.40%. This is quite a decrease when compared to previous tests. However, such decline in performance can be associated to the temporal nature of the data, since through all the years in the dataset, writing methodologies and overall data handling principles have changed. This demonstrates the importance of keeping the dataset updated, in order to achieve optimal results.

4.6 Summary

Regarding the models implemented, we can see from the results in Table 4.1 that the classifier displaying best results was Logistic Regression, followed closely by the Multi-Layer Perceptron and also Support Vector Machines.

The lowest scores between classifiers (not taking into consideration the Baseline) were observed in both Naive Bayes – for which said results can be justified by the simplicity of the classifier and its assumptions – and the sequential models, which suffer from the depth of the vocabulary against its breadth, consisting in documents within the same context and with low ambiguity and the fact that the

designed models could take advantage of more samples to train on.

Looking into the dataset imbalance it might be valuable to use a balancing method such as oversampling. While this mechanism displayed slightly worse results than for the imbalanced set, it is a good alternative for a system in which we are looking for higher recall for the minority classes, especially given the fact that such classes are the source of biggest confusion for our classifiers.

Chapter 5

Conclusions

This dissertation presented multiple Machine Learning classifiers to be used as an alternative to the manual distribution classification performed currently at the Portuguese Navy. This chapter provides an overview over the main contributions and presents possible ideas for future work.

5.1 Achievements/Contributions

This dissertation explored the correspondence distribution panorama taking place currently in the Portuguese Navy. This process is done by hand, prone to human error and subjectivity, along with the extra time consumption required to perform the task. These effects are undesirable and can hinder the productivity of a large organization such as the Portuguese Navy.

This work tested several classifiers in multiple settings, in order to select the best to perform this task. Analysis of the results revealed the major liabilities linked to this task, such as the imbalance between classes, with one of them being in much minor representation, and the temporal factor of the data which was demonstrated by the decrease in performance, when evaluating results in a set separated by belonging years.

This work explored the influence of document representation and sampling methods to find the classifier that outperformed the others. With this goal in mind, this dissertation presents a classification method for automatic correspondence distribution using Logistic Regression and the principles of Binary Relevance for Multi-label Classification with the aid N-gram models that displayed an average accuracy of 82.02% over seven distinct labels, correctly predicting all seven labels for 35.40%. Regarding the imbalance of the dataset, an alternative using Oversampling was also presented, displaying little accuracy decrease when compared to the imbalanced set and a significant increase in recall.

5.2 Future Work

For future work, there are various ideas that could help provide better results for this task and improve the classification performance. For starters, there is some noise being introduced through the OCR,

producing illegible words that affect the quality of the dataset.

After that, regarding data pre-processing, it would be a good idea to try Stemming to give more weight to context and less to specific words and to implement Named Entity Recognition, since when looking into the features with more weight to classification, some classifiers presented personal names. It would be a good idea to replace said names by the person's rank within the Navy in order to prevent biases towards rotating roles. Besides that, it would be good to have more documents to train on, in order to fully experiment with more complex models which for this task were not the adequate choice.

Bibliography

[1] A. Madasu and V. A. Rao. Sequential learning of convolutional features for effective text classification. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5657–5666. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1567. URL https://doi.org/10.18653/v1/D19-1567.

[2] F. Dernoncourt. *Sequential short-text classification with neural networks*. PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2017. URL http://hdl.handle.net/1721.1/111880.

[3] M. M. Bittencourt, R. M. Silva, and T. A. Almeida. Ml-mdltext: A multilabel text categorization technique with incremental learning. In *8th Brazilian Conference on Intelligent Systems, BRACIS 2019, Salvador, Brazil, October 15-18, 2019*, pages 580–585. IEEE, 2019. doi: 10.1109/BRACIS.2019.00107. URL https://doi.org/10.1109/BRACIS.2019.00107.

[4] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–7. IEEE, 2015. doi: 10.1109/IJCNN.2015.7280624. URL https://doi.org/10.1109/IJCNN.2015.7280624.

[5] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognit.*, 37(9):1757–1771, 2004. doi: 10.1016/j.patcog.2004.03.009. URL https://doi.org/10.1016/j.patcog.2004.03.009.

[6] H. Shao, G. Li, G. Liu, and Y. Wang. Symptom selection for multi-label data of inquiry diagnosis in traditional chinese medicine. *Sci. China Inf. Sci.*, 56(5):1–13, 2013. doi: 10.1007/s11432-011-4406-5. URL https://doi.org/10.1007/s11432-011-4406-5.

[7] H. Chougrad, H. Zouaki, and O. Alheyane. Multi-label transfer learning for the early diagnosis of breast cancer. *Neurocomputing*, 392:168 – 180, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.01.112. URL http://www.sciencedirect.com/science/article/pii/S0925231219304710.

[8] A. Chan and A. A. Freitas. A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 27–34. ACM, 2006. doi: 10.1145/1143997.1144002. URL https://doi.org/10.1145/1143997.1144002.

[9] J. Liu, W. C. Chang, Y. Wu, and Y. Yang. Deep learning for extreme multi-label text classification. *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124, 2017. doi: 10.1145/3077136.3080834.

[10] E. Gibaja and S. Ventura. A tutorial on multilabel learning. *ACM Computing Surveys*, 47(3), 2015. ISSN 15577341. doi: 10.1145/2716262.

[11] D. W. Castro, E. Souza, D. Vitório, D. Santos, and A. L. Oliveira. Smoothed n-gram based models for tweet language identification: A case study of the brazilian and european portuguese national varieties. *Applied Soft Computing*, 61:1160 – 1172, 2017. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2017.05.065. URL http://www.sciencedirect.com/science/article/pii/S1568494617303459.

[12] C. S. Lim, K. J. Lee, and G. C. Kim. Multiple sets of features for automatic genre classification of web documents. *Information Processing Management*, 41(5):1263 – 1276, 2005. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2004.06.004. URL http://www.sciencedirect.com/science/article/pii/S0306457304000676.

[13] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng. News impact on stock price return via sentiment analysis. *Knowledge-Based Systems*, 69:14 – 23, 2014. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2014.04.022. URL http://www.sciencedirect.com/science/article/pii/S0950705114001440.

[14] M. F. R. A. Bakar, N. Idris, L. Shuib, and N. Khamis. Sentiment analysis of noisy malay text: State of art, challenges and future work. *IEEE Access*, 8:24687–24696, 2020. doi: 10.1109/ACCESS.2020.2968955. URL https://doi.org/10.1109/ACCESS.2020.2968955.

[15] T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206 – 10222, 2009. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2009.02.037. URL http://www.sciencedirect.com/science/article/pii/S095741740900181X.

[16] G. A. Rodrigo. Projeto: Identificação e Classificação de Entidades Mencionadas e Eventos em Documentos da Marinha. 2020. Master Thesis. IST, UL.

[17] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104, 2012. ISSN 00313203. doi: 10.1016/j.patcog.2012.03.004.

[18] M. L. Zhang, Y. K. Li, X. Y. Liu, and X. Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018. ISSN 20952236. doi: 10.1007/s11704-017-7031-7.

[19] N. Spolaôr, E. A. Cherman, M. C. Monard, and H. D. Lee. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151, 2013. ISSN 15710661. doi: 10.1016/j.entcs.2013.02.010. URL http://dx.doi.org/10.1016/j.entcs.2013.02.010.

[20] J. Li, F. Alzami, Y. Gong, and Z. Yu. A multi-label learning method using affinity propagation and support vector machine. *IEEE Access*, 5:2955–2966, 2017. doi: 10.1109/ACCESS.2017.2676761. URL https://doi.org/10.1109/ACCESS.2017.2676761.

[21] S. M. Rendón, D. H. Peluffo-Ordóñez, and G. Castellanos-Domínguez. support vector machine-based aproach for multi-labelers problems. In *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*, 2013. URL http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-118.pdf.

[22] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Mach. Learn.*, 85(3):333–359, 2011. doi: 10.1007/s10994-011-5256-5. URL https://doi.org/10.1007/s10994-011-5256-5.

[23] C. Zhang, Z. Yu, H. Fu, P. Zhu, L. Chen, and Q. Hu. Hybrid Noise-Oriented Multilabel Learning. *IEEE Transactions on Cybernetics*, pages 1–14, 2019. ISSN 2168-2267. doi: 10.1109/tcyb.2019.2894985.

[24] M. Zhang and Z. Zhou. A k-nearest neighbor based algorithm for multi-label classification. In X. Hu, Q. Liu, A. Skowron, T. Y. Lin, R. R. Yager, and B. Zhang, editors, *2005 IEEE International Conference on Granular Computing, Beijing, China, July 25-27, 2005*, pages 718–721. IEEE, 2005. doi: 10.1109/GRC.2005.1547385. URL https://doi.org/10.1109/GRC.2005.1547385.

[25] H. Jain, Y. Prabhu, and M. Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu:935–944, 2016. doi: 10.1145/2939672.2939756.

[26] J. Read and F. Pérez-Cruz. Deep learning for multi-label classification. *CoRR*, abs/1502.05988, 2015. URL http://arxiv.org/abs/1502.05988.

[27] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 995–1000. IEEE Computer Society, 2008. doi: 10.1109/ICDM.2008.74. URL https://doi.org/10.1109/ICDM.2008.74.

[28] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4701 LNAI, pages 406–417, 2007. ISBN 9783540749578. doi: 10.1007/978-3-540-74958-5_38.

[29] D. Jurafsky and J. H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. ISBN 9780135041963. URL https://www.worldcat.org/oclc/315913020.

[30] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[31] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[32] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN 9781999579517. URL https://books.google.pt/books?id=0jbxwQEACAAJ.

[33] T. Pranckevičius and V. Marcinkevičius. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, 5(2):221–232, 2017. ISSN 2255-8950. doi: 10.22364/bjmc.2017.5.2.05.

[34] D. Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012. ISBN 0521518148.

[35] J. Hovold. Naive bayes spam filtering using word-position-based attributes and length-sensitive classification thresholds. In S. Werner, editor, *Proceedings of the 15th Nordic Conference of Computational Linguistics, NODALIDA 2005, Joensuu, Finland, May 2005*, pages 78–87. University of Joensuu, Finland, 2005. URL https://www.aclweb.org/anthology/W05-1712/.

[36] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010. ISBN 978-0-13-207148-2. URL http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0136042597,00.html.

[37] M. Sorower. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, pages 1–25, 2010. URL http://people.oregonstate.edu/~sorowerm/pdf/Qual-Multilabel-Shahed-CompleteVersion.pdf.

[38] D. Ganda and R. Buch. A Survey on Multi Label Classification. *Recent Trends in Programming Languages*, 5(August), 2018.

[39] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009. ISSN 08856125. doi: 10.1007/s10994-009-5127-5.

[40] ML Universal Guides text classification. https://developers.google.com/machine-learning/guides/text-classification/. Accessed: 2020-12-31.

[41] X. Liu, J. Wu, and Z. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. Part B*, 39(2):539–550, 2009. doi: 10.1109/TSMCB.2008.2007853. URL https://doi.org/10.1109/TSMCB.2008.2007853.