FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Intelligent Analysis of Complaints

Luís Fernando Araújo da Silva Vilar Barbosa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Henrique Daniel de Avelar Lopes Cardoso

Second Supervisor: Gil Filipe da Rocha

July 18, 2019

Intelligent Analysis of Complaints

Luís Fernando Araújo da Silva Vilar Barbosa

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Jorge Manuel Gomes Barbosa
External Examiner: Brígida Mónica Teixeira Faria
Supervisor: Henrique Daniel de Avelar Lopes Cardoso

July 18, 2019

Abstract

In the last few years, several institutions, public and private, started supplying several services electronically. Some of these institutions provide the possibility of making complaints electronically. The collection and analysis of customer feedback are important to allow organizations adapting their offerings, providing better services.

Some public institutions are responsible for analyzing citizen complaints. These complaints can come in large numbers and are mainly composed of free text. Due to the amount of complaints received, it is necessary to create some automatic way to process them, at least to some extent.

Usually, electronic complaint management systems have options in a contact form to indicate to which domain category a complaint is related so it is easier to handle the complaint and, if necessary, allocate the complaint to a department or person. This is a good way of facilitating the management of complaints when we only need to handle electronic complaints and when we already have a way to indicate categories on the complaints management system. However, when we have complaints written by hand that need to be moved to an electronic format, or complaints electronically sent as free text by e-mail, other ways of analyzing the text become necessary.

This thesis focuses on the task of automatically or semi-automatically identifying economic activities and infractions (or a generalization thereof) present in complaints submitted to the Portuguese Economic and Food Safety Authority (ASAE), employing natural language processing (NLP) and machine learning (ML) techniques for Portuguese, a language with few resources in terms of NLP. Different features are employed and analyzed to discover which ones give the best results using different machine learning algorithms. The thesis aims to propose an easily adaptable system capable of classifying complaints, but that can be adjusted to perform the same tasks on other types of textual data.

The work contributed for the creation of a component to be integrated into a platform for the management of economic agents, complaints and inspections. This component will be a good aid in the process of analysis and classification of complaints that until now has been done manually, with the benefits of speeding up the analysis and reducing the number of cases where the classification is wrong.

Keywords: Machine learning, Text-mining, Text categorization, Natural language processing, User-generated text, Complaint analysis, Economic activity prediction, Infraction prediction

Resumo

Nos últimos anos, diversas instituições, públicas e privadas, começaram a fornecer vários serviços eletronicamente. Algumas destas instituições providenciam a possibilidade de realizar reclamações e/ou denúncias eletronicamente. A coleção e análise da opinião do cliente são importantes para permitir que as organizações adaptem as suas ofertas, oferecendo melhores serviços.

Algumas instituições públicas são responsáveis por analisar denúncias dos cidadãos. Estas denúncias podem ser recebidas em quantidades gigantescas e são compostas, principalmente, por texto livre. Devido à quantidade de denúncias recebidas, é necessário criar alguma forma automática de processá-las, pelo menos até certo ponto.

Normalmente, os sistemas eletrónicos de gestão de denúncias têm uma opção no formulário para indicar a que categoria uma denúncia está relacionada, tornando mais fácil a gestão da denúncia e, se necessário, alocar a denúncia a um departamento ou pessoa. Esta é uma boa forma de facilitar a gestão de denúncias quando só precisamos lidar com denúncias eletrónicas e quando já temos uma maneira de indicar categorias no sistema de gestão de denúncias, mas, quando temos denúncias escritas à mão que precisam de ser transferidas para um formato eletrónico e denúncias enviadas eletronicamente sob a forma de texto livre, como por *e-mail*, outras formas de analisar o texto são necessárias.

Esta dissertação foca-se na tarefa de identificar de forma automática ou semiautomática as atividades económicas e infrações (ou uma generalização delas) presentes nas denúncias submetidas à Autoridade de Segurança Alimentar e Económica (ASAE) Portuguesa, recorrendo a técnicas de processamento de linguagem natural (NLP) e aprendizagem máquina (ML) para Português, uma língua com poucos recursos em termos de NLP. Diferentes recursos são empregues e analisados para descobrir quais fornecem os melhores resultados usando diferentes algoritmos de aprendizagem máquina. A dissertação tenciona propor um sistema facilmente adaptável capaz de classificar denúncias, mas que pode ser ajustado para realizar as mesmas tarefas em outros tipos de dados textuais.

O trabalho contribuiu para a criação de um componente a ser integrado numa plataforma de gestão de agentes económicos, denúncias e fiscalizações. Este componente será uma boa ajuda no processo de análise e classificação das denúncias que até agora era feito de forma manual, tendo como benefícios a maior celeridade na realização da análise e a redução do número de casos em que a classificação é errada.

Palavras-chave: Machine learning (aprendizagem máquina), Text-mining (mineração de texto), Text categorization (categorização de texto), Natural language processing (processamento de linguagem natural), User-generated text (Texto gerado pelo utilizador), Complaint analysis (análise de denúncias), Economic activity prediction (Previsão da atividade económica), Infraction prediction (previsão de infração)

Acknowledgements

I would like to show my gratitude to my supervisors, the professor Henrique Lopes Cardoso and Gil Filipe da Rocha, and to João Filgueiras, colleague on the IA.SAE project, which allowed me to improve this thesis with their comments, suggestions and support.

Equally, I would like to manifest my gratitude to the members of the jury, Henrique Lopes Cardoso, Mónica Faria and Jorge Barbosa, for their contribution.

I would also like to show my gratitude to my parents which supported me along this journey and through my life and, to finalize, I would like to show my gratitude to my friends, colleagues and professors that helped me along my academic activities.

Thank you very much to everyone.

Luís Barbosa

"We all have our time machines.
Some take us back, they're called memories.
Some take us forward, they're called dreams."

Jeremy Irons

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AI	Artificial Intelligence
ASAE	Autoridade de Segurança Alimentar e Económica (Economic and Food Safety Authority)
CPV	Category-Pivoted Categorization
DPC	Document-Pivoted Categorization
IA.SAE	Inteligência Artificial na Segurança Alimentar e Económica (Artificial Intelligence in Economic and Food Safety)
KPI	Key Performance Indicator
LDA	Latent Dirichlet Allocation
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores (Artificial Intelligence and Computer Science Laboratory)
ML	Machine Learning
NB	Naive Bayes
NLP	Natural Language Processing
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machines
TF	Term Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
VSM	Vector Space Model

Chapter 1

Introduction

Several countries have public administration institutions that provide public services electronically. Such institutions are responsible for processing citizen requests, also performed by electronic means, often materialized through email contacts or by filling-in contact forms. In specific types of public institutions, such as those in charge of enforcing compliance of citizens or economic agents, a significant number of such requests are in fact complaints that need to be appropriately dealt with.

The quantity of complaints received can easily reach the thousands in a short amount of time, depending on the size of the country/administrative region. *Autoridade de Segurança Alimentar e Económica* (ASAE) is the Portuguese authority responsible for food safety and economic surveillance. It is responsible for the inspection of commercial entities and for evaluating and communicating risks in the economic chain. ASAE is a complex organism that needs to manage a large quantity of information, from e-mails that inform that a given store is performing a specific promotion to complaints sent by large organizations or even individuals.

ASAE receives more than 20 thousand complaints annually and, in a period of 10 years, approximately 30% of the complaints have been found not to be in the jurisdiction of ASAE; the rest are sent to the ASAE operational units. All of the information gathered by ASAE cannot be efficiently managed by human operators without computational help and, due to this reason, the IA.SAE (Artificial Intelligence in Economic and Food Safety) project has been started in collaboration with *Laboratório de Inteligência Artificial e Ciência de Computadores* (LIACC) to streamline some of the processes performed inside ASAE. One of the obstacles to do it effectively is the fact that contact forms typically include free-form text fields, bringing high variability to the quality of the content written by citizens.

This thesis focuses on a specific module of the IA.SAE project that will be better contextualized below. It aims to use natural language processing (NLP) and machine learning (ML) techniques to extract information from free text written in Portuguese, which is a low-resourced language in terms of NLP.

1.1 Context

ASAE receives numerous complaints related to potential infractions of different economic agents from the various sectors of activity in which it operates. Although carried out by electronic means, such complaints are currently hand-checked, in terms of their content, classification and prioritization, which requires an unsustainable amount of human resources. To change this situation, allowing for a better and faster organization of the complaints, the IA.SAE project has been created.

The IA.SAE project is being developed at Faculdade de Engenharia da Universidade do Porto, more specifically by LIACC research laboratory. The project is divided into several modules that can be made independently. The different modules will be presented below. This thesis will focus on the first module, the classification of the complaints, trying to use text mining algorithms to achieve this objective. The main objectives of this project are to develop models of risk analysis and selection of economic agents to be monitored using the data present on the databases that already exist and that will be created at ASAE.

The models of risk analysis should use the information available in ASAE's databases to improve prevention in the areas of food security and economic surveillance and use techniques of fusion and extraction of information, machine learning, optimization, data and text mining, simulation of the behavior of the different economic agents and intelligent visualization of the information.

A brief overview of them will be made to allow a better comprehension of the project. Although the IA.SAE project has a few more activities, Figure 1.1 presents an overview of the relationship between the project activities, which are related to the modules that will be presented below. Only the core activities are presented, excluding those related with project management.

- Module 1: Smart Prioritization and Separation of Complaints
  - Extract the most important information from electronic complaints.
  - Crossing of information regarding the complaint, economic agent and its history from multiple sources.
  - Intelligent prioritization of electronic complaints.
  - Intelligent separation of complaints by type/area.
  - Continuous system learning with information/corrections of human operators for continuous improvement of their operation.

  This first module is the one that will be most emphasized along this document because the aim of this thesis is the resolution of the items presented inside this module.

- Module 2: Smart Inspection of Balances and Promotions
  - Intelligent analysis of emails with notifications of balances with semiautomatic extraction of relevant information.

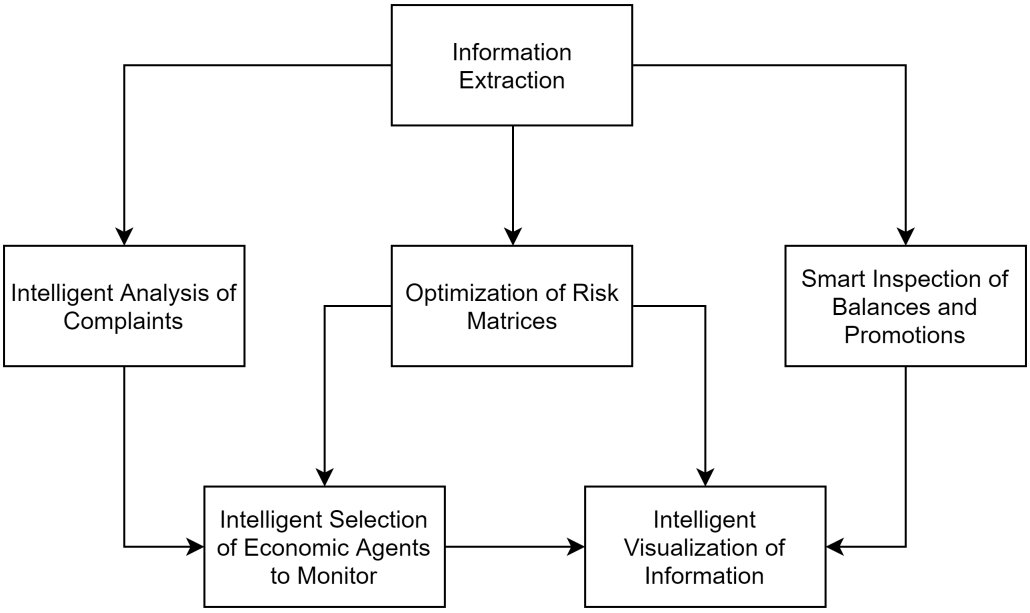

Figure 1.1: Core IA.SAE Activities

  - Generation of reports and alarms for undue balances and finished balances.
  - Daily/weekly generation of routes/inspection plans for georeferenced balances.
  - Continuous learning of the system with new information on the results of the performed verifications of balances for continuous improvement of its functioning.

- Module 3: Smart Optimization of Risk Matrices
  - Generation of global risk matrix based on the consumption volume, default rate, product/service and other factors.
  - Optimization of risk matrices by maximizing the correlation between the risk and the result of the inspections.
  - Continuous learning of the system with information of the results of each inspection for continuous improvement of its operation.

- Module 4: Intelligent Selection of Economic Agents to Monitor
  - Fusion of information regarding each economic agent, its history and current information from multiple sources.
  - Creation of models of economic agents and intelligent simulation of the behavior of these economic agents based on the models created.
  - Intelligent selection of economic agents to be monitored based on models and simulations.
  - Generation of georeferenced plans and inspection routes based on the selection of economic agents to be supervised.

  - Continuous learning of the system, with new information on the results of the inspections of economic agents carried out, for continuous improvement of its operation.

- Module 5: Intelligent Visualization of KPIs and Georeferenced Information
  - Generation of flexible inspection routes.
  - Interactive visualization of maps and routes/inspection paths.
  - Generation of appropriate Key Performance Indicators (KPIs).
  - Intelligent visualization of KPIs with decision support measures.
  - Provision of a global interface for the management and configuration of the entire IA.SAE system.

1.2 Motivation and Objectives

As said earlier, ASAE receives numerous complaints. For every complaint, ASAE staff extracts the target economic agent, decides the economic activities to which the complaint is related, decides the possible infractions suggested in the complaint and, finally, decides the entity with competence.

The extraction of the target economic agent is not in the scope of this thesis, so the first problem is the assignment of the economic activities, because, at the moment, there are 360 economic activities available to choose from. Having in mind that this task is being performed manually and that more than 20 thousand complaints are received annually, it is necessary a large amount of human resources to employ the task of analyzing the complaints.

After obtaining the economic activity, it is necessary to identify which infractions might be present in the complaint. Depending on the infractions present in the complaint, it is possible to identify if ASAE is the authority with competence to manage the complaint.

The information provided in the complaints is a negative form of feedback from the user experience [OTB+14] which can be organized based on the type of problem indicated or the economic area in which it falls, among others [CGR18]. Organizing the complaints by categories allows the division of complaints, which will increase the efficiency of their processing by ASAE staff. To extract the relevant information from a complaint, in the form of free text, we aim to employ natural language processing, information extraction and machine learning techniques [APA+17]. The categories can be obtained using different procedures [CGR18] and allow to filter the information obtained [Seb02].

This thesis focuses on the intelligent and semi-automatic analysis of complaints received by electronic means, carrying out their cataloging [APA+17, Seb02], the identification of the type of infraction presented in the complaint (if any) and the identification of the entity with competence to handle the complaint. The aim is to obtain a prototype of an intelligent system for the analysis of electronic complaints. With this, it is expected the creation of classification models that will be

used in conjunction with other modules of the IA.SAE project to build a prioritization model for a real application of high public and economic interest that may be usable for other purposes.

For the development of the work, it is intend to resort to machine learning and text mining techniques [APA+17], especially focused on the exploration of text data, for the creation of a prototype that allows classifying the complaints made to ASAE. To train the prototype, ASAE data will be used. In addition to a learning model that correctly classifies complaints, it is intended to integrate it into a functional prototype that serves as proof of concept.

1.3 Research Questions

This thesis aims at answering the following research questions for the task of predicting the economic activity and the task of predicting the infractions:

- Which NLP techniques should be used to preprocess user-generated complaints?
  - Several preprocessing techniques can be used, such as tokenization, lemmatization, stemming, removal of stop words, removal of words with a specific syntactic meaning (adjectives, nouns, verbs), substitution of synonyms and others. These techniques are applied during preprocessing and the aim is to comparatively test the benefits of using different preprocessing techniques in view of the target task.
- Which data representation techniques obtain best results to categorize complaints?
  - There are several data representation techniques which might be used, such as count and TF-IDF, between others. These techniques are applied during feature extraction and the aim is to comparatively test the benefits of using different data representation techniques in view of the target task.
- Which text categorization algorithms are the best to categorize complaints and, for each one, which features should be extracted?
  - There are several algorithms that can be used to categorize complaints. Each of them might obtain better or worst results compared to others using the same input. The aim is to comparatively test the benefits of using different categorization algorithms and to comparatively test the benefits of different feature selection techniques in view of the target task.
- How effective is the use of text categorization algorithms as a recommendation to humans by providing labels in the form of a ranking?
  - Some text categorization algorithms can provide the probability associated to a given label. If the probability for all possible labels is available, it is possible to provide a ranking from which the user can choose an option. The aim is to comparatively test the benefits of using rankings with different sizes in view of the target task.

1.4 Contributions

This thesis intended to create highly accurate models that can be integrated into a functional prototype that serves as proof of concept, being part of a framework easily adaptable to other use cases and project modules. The main contributions are:

- A model that predicts with a high accuracy score the economic activity class of a complaint.
- A model that predicts with an acceptable accuracy score the most severe type of infraction present in a complaint.
- The integration of the models in the general platform of the project.
- A paper submitted to the 7th International Conference on Statistical Language and Speech Processing, entitled "Automatic Identification of Economic Activities in Complaints" and present in Appendix B.

The experiments performed can provide insights about which techniques might be or not pursued in similar tasks and suggest new experiments that might improve the classification tasks and the state of the art.

1.5 Structure of the Thesis

This thesis has 7 more chapters.

In Chapter 2, the techniques and concepts behind text mining and text categorization are approached.

In Chapter 3, the state of the art is described. Unlike Chapter 2, this chapter will only approach the state of the art related to the analysis of complaints. In the beginning of the chapter, techniques used for complaint analysis will be mentioned. Immediately after that, an overview of possible support technologies will be given and followed by a few conclusions on the overall state of the art.

In Chapter 4, the datasets used in this work are described and an overview of the structure of the economic activities and infractions is given.

In Chapter 5, it is indicated which preprocessing and feature extraction techniques were applied to the data and the libraries used. It is indicated why they were chosen and a brief explanation is given on how some problems related to the efficiency of the prototype and its usability were solved.

In Chapter 6, several different experiments that were performed regarding the prediction of economic activities are described.

In Chapter 7, experiments that were performed regarding the prediction of the most severe infraction category are described.

To conclude, in Chapter 8 it will be presented the overall conclusions and future work.

Chapter 2

Background

This thesis aims to address the task of automatic or semi-automatic analysis of complaints. Before introducing the state of the art related to the analysis of complaints, it is important to understand the techniques and concepts that allow the automatization of the analysis of the free text present in the complaints.

Artificial Intelligence (AI) is the research field that encompasses Machine Learning (ML), which is subdivided into other fields such as Text Mining (TM). The latter studies the automatic analysis and interpretation of text.

Artificial intelligence is a concept that refers to allowing machines to perform tasks that we, as humans, consider smart and difficult to program [RND09]. Machine learning is part of artificial intelligence and is a concept that refers to allow machines to learn how to perform tasks by them self [Mit97]. This type of tasks, specially orientated to text, will be presented across this chapter.

2.1 Text Mining

Text mining or knowledge discovery from text was introduced by Feldman and Dagan [FD95] and refers to the process of extracting information from either structured or unstructured text resources [APA+17] through the discovery of unknown and, probably, non-trivial patterns [ZJN18].

For a better comprehension of the topics that will be explored throughout the chapter, a few introductory definitions will be given.

Information Retrieval (IR): Information retrieval is the process of retrieving usable information from information resources (usually in the form of documents) that present the information in an unstructured or semi-structured manner. It is mostly focused on finding information and retrieving it, not in analyzing information and finding hidden patterns. [APA+17, MRS09]

Natural Language Processing (NLP): Natural language processing, a sub-field of computer science, artificial intelligence and linguistics, aims to study techniques that allow computers to

process and understand natural language and retrieve meaningful information from the processed data. [APA+17, MS99]

Information Extraction from text (IE): Information extraction from text is the activity of automatically extracting information from unstructured or semi-structured text documents [APA+17, AJ18]. Usually, it is a starting point for other text mining algorithms [APA+17].

Unsupervised Learning Methods: Unsupervised learning techniques try to find hidden structure out of unlabeled data [APA+17, MS99, Bis06]. This type of techniques use proximity metrics to create groups of documents. Clustering and topic-modeling are two unsupervised learning algorithms used with text data. Clustering corresponds to take a collection of documents and group those of the same type, i. e., those that are similar to a cluster and the others to other clusters [APA+17, MS99, Bis06]. This technique allows to incrementally make better clusters because the clusters are updated whenever new documents are given. Topic-modeling has a different behavior because instead of assigning a document to a cluster, it gives a probability of a given document belong to a given cluster, existing a probability of a document belong to any cluster [APA+17].

Supervised Learning Methods: Supervised learning methods are techniques that use labeled data to infer a function that allows inferring predictions on new data. [APA+17, Bis06] There are a lot of supervised learning methods and some of the most known will be referred along this chapter.

Now that a few introductory definitions have been given, we are going to introduce the text preprocessing phase and some data representation techniques that are used the for feature extraction phase. These two phases occur before the execution of any classifier.

2.1.1 Text Preprocessing

On the text preprocessing phase, there are a few tasks that need to be performed to improve and facilitate the usage of the classifiers. With each of the following tasks, it is possible to achieve better results when using the classifiers, so although it is not necessary to perform all of them, the execution of each one in conjunction with the others might yield better results.

This phase is fundamental to transform free text in a structured format (through tokenization) that can be processed by computers, cleaning out irrelevant words (filtration) and, in some cases, adding additional information, as occurs by the placement of part-of-speech taggers. This list is not an extensive one, there are many ways of text preprocessing, these are the most used.

Tokenization: Tokenization is the activity of dividing a stream of characters (a text) into smaller pieces (words/phrases) called tokens [APA+17, MRS09]. Exemplifying, the expression "A good person." can be divided in: "A", "good", "person." or "A", "good", "person", ".".

Lemmatization: Lemmatization is the activity of analyzing morphologically a word. For this to occur, it is necessary to group all the words that have the same meaning and root and match them to only one definition. [APA+17, MRS09] Exemplifying, the lemma of "moved" is "move".

Stemming: Stemming is the activity of obtaining the root of a word [APA+17, ZJN18, MRS09]. Exemplifying, the stem of the lemma "move" is "mov".

Filtering: Filtering is the activity of removing unnecessary pieces of data (tokens or parts of tokens) from the whole data. Usually, these pieces do not give very important information. [APA+17, ZJN18] Punctuation is removed at this stage.

2.1.2 Data Representation

Usually, the text is divided into tokens, a few operations are performed on the tokens, like lemmatization, and, then, a matrix of features is created. The matrix of features is based on the Vector Space Model (VSM) where each row corresponds to one example and each column corresponds to one feature extracted.

To explain how a VSM works, we need to define a few terms and variables. Let us assume a collection of documents $D = \{d_1, d_2, ..., d_n\}$, a collection of words $W = \{w_1, w_2, ..., w_n\}$ and a collection of numbers $N = \{n_1, n_2, ..., n_n\}$, where w_i is the word corresponding to n_i and n_i is the number corresponding to w_i.

The most common way of representing the features of a document is through a vector of numbers. To simplify, let us assume that each feature is a word w_i of the document. In VSM, each word w_i has a value n_i which can be defined using different encoding techniques, such as: binary encoding, term frequency(TF) and term frequency-inverse document frequency (TF-IDF). A *binary allocation* corresponds to assign 0 to n_i when w_i is not present in document d_j, or assign 1 otherwise. A *term frequency allocation* corresponds to calculate the frequency of the word w_i in the document d_j and assign the frequency to n_i. A *term frequency-inverse document frequency allocation* corresponds to use the corresponding formula to calculate a value which indicates the importance of the word w_i on all documents and assign the value to n_i. [APA+17]

Another technique is the use of word embeddings [Gol15] to create a vector for each document. Usually, when this technique is used, the embedding vector associated to each token is obtained and all the embedding vectors obtained for all features are summed (assuming that all vectors have the same dimensionality) to obtain the final vector [Gol15]. There are several pre-training methods available that provide models to obtain embeddings from which MUSE (Multilingual Unsupervised and Supervised Embeddings) [1], fastText [2], polyglot [3], Bidirectional Encoder Representations from Transformers (BERT) [4] and ELMo [5] are only a few.

After obtaining a vector populated with the values that represent the features of the document, it is possible to compare the vector of features of the document d_i with the vector of features of the document d_j and obtain a similarity measure [APA+17] that classifiers use to distinguish classes.

[1] https://github.com/facebookresearch/MUSE
[2] https://github.com/Babylonpartners/fastText_multilingual
[3] https://polyglot.readthedocs.io/en/latest/CLI.html
[4] https://github.com/google-research/bert
[5] https://github.com/HIT-SCIR/ELMoForManyLangs

2.2 Text Categorization

Text categorization is the activity of assigning a boolean value indicating if a document belongs to a given category. In mathematical form, given a collection of documents $D = \{d_1, d_2, ..., d_n\}$ and a collection of categories $C = \{c_1, c_2, ..., c_m\}$ where n and m represent the total of documents and the total of categories, respectively, text categorization corresponds to assign a boolean value to a pair $\{d_i, c_i\}$ indicating if the document is related to the category or not. [Seb02]

Thus, text categorization algorithms try to find an unknown function, which is called the classifier, that allows generalizing the retrieval of the boolean value for documents that the algorithm never analyzed [Seb02].

The semantic analysis of text is a subjective notion where someone indicates the category of a document and, because it is subjective, it is possible to occur a phenomenon of inter-indexation where two people give two different classifications to the same text [Seb02].

2.2.1 Types of Text Categorization

Single-Label Versus Multilabel Text Categorization

There are different restrictions that can be imposed when classifying text. Being k the number of defined categories, c the number of categories to which a document must belong and b a constant, the following restrictions can be imposed:

- $c = k + b$
- $c <= k + b$
- $c >= k + b$
- $c < k + b$
- $c > k + b$

If c = 1 and k is between 2 and |C|, a single-label text categorization, also known as non-overlapping categorization, is being performed [Seb02].

If c and k are between 2 and |C|, a multilabel text categorization, also known as overlapping categorization, is being performed [Seb02].

A special and interesting case is the binary categorization where there are only two categories and the category of a document can be *cat* or $\neg cat$ [Seb02].

Category-Pivoted Versus Document-Pivoted Text Categorization

When using a text classifier, it is important to know if we want to know all the categories to which a document belongs or if we want to know all the documents that belong to each category. If we want to know the categories to which a document belongs, we want to perform a document-pivoted categorization (DPC). By the other side, if we want to know which documents belong

to a category, we want to perform a category-pivoted categorization (CPC). This distinction is important when the complete set of categories or documents is not available from the start and it will be relevant for the choice of the classifier to use. [Seb02]

Document-pivoted categorization is best suitable when new documents may be added to the list of documents, not being necessary to rerun the classifier after it has categorized all the documents already present in the list.

Category-pivoted categorization is best suitable when new categories may be added to the list of categories, not being necessary to rerun the classifier after it has indicated all the documents that belong to a category.

Depending on the classifier, it might be necessary to choose between document-pivoted categorization and category-pivoted categorization, but, in most cases, the classifiers are capable of working using both modes. [APA+17]

Specific Categorization Versus Ranking Categorization

It is possible to perform a specific categorization where the automated text categorization mechanism gives a specific category or set of categories to a given document. A specific categorization is used when a fully automated mechanism is implemented.

There are cases where a human operator is in charge of assigning a category to the document, either because there is no categorization algorithm capable of it or because the categorization is critical and a high rate of accuracy is needed. In these cases, instead of categorizing the document, the classifier shows a ranking indicating which categories have a higher probability and lower probability. This would be a great help for the human operator in charge of taking the decision, reducing its decision time and allowing him to make better decisions. [Seb02]

2.2.2 Some Applications of Text Categorization

Text categorization can have several applications. It can be used to allow the search for a given topic in databases like Wikipedia through the categorization of the different pages, indexing the ones that belong to the same categories. It can be used to create groups of Web pages that belong to the same categories and, like will be approached below, even a hierarchical categorization can be made. It can be used to find other documents that are similar, helping on the creation of programs that check for plagiarism. In the following subsections, a few generic applications will be described.

Automatic Indexing for Boolean Information Retrieval Systems

Automatic indexing for boolean information retrieval systems is used on databases that contain documents of specialized subjects. In boolean information retrieval systems, key words or key sentences are assigned to each document [Seb02]. Whenever a query is made, the entries are checked and, if a match occurs, the document is retrieved.

The indexing on these systems can resort to humans to perform a manual indexation, but text categorization can be used if a specific vocabulary is set for the automatic indexing mechanism, being used for all documents. [Seb02]

Usually, this type of indexing resorts to a document-pivoted text categorization [Seb02] because new documents can be added all the time, but the same vector of features (words/expressions) can be used.

Document Organization

Document organization is a typical text categorization problem if there is no specific format for the content and metadata of files.

In the case of newspapers, a given news can be categorized as politics, sports, finances and others. The content of the news is free text. Text categorization is a good approach to automatically classify the news, grouping the ones belonging to each category. [Seb02]

Text Filtering

Text filtering is the task of using the classification of documents to decide if they should be maintained/received by the system. The classification must be done before performing the decision and can resort to text categorization techniques. An example presented by Sebastiani [Seb02] is a newsfeed which receives news from a news agency and delivers only those that belong to some categories to a newspaper. In this case, the filtering system should not deliver news that belong to categories to which the consumer is not interested. [Seb02]

Word Sense Disambiguation

Word sense disambiguation is the task of using the category of the document to understand the meaning of a word that can have an ambiguous meaning, i. e., a different meaning depending on the context. An example is the word “bank”, which can be an object used to sit or a financial institution. Word sense disambiguation is very important for tasks like natural language processing by allowing the distinction of the situation, which might be crucial on tasks like text categorization [Seb02]. For instance, an automatic text categorization system could easily see the word “bank” appearing in a news with the sense of an object to sit, but allocate the document in which it occurs to the category of financial news.

Hierarchical Categorization of Web Pages

The hierarchical categorization of Web pages is applied, for instance, on many Internet portals. These websites show a list of categories organized in a hierarchic format and, inside each category, they present links to other Web sites that have content related to that category. The use of categories instead of searching the Web using a search engine can be easier when the user does not know exactly which terms to use for the query. Also, it can help the user situate its search, so later he

can use the search engine. Since the number of Web pages available on the Web is very large and is mainly free text, the use of text categorization is a good option to classify different Web pages. [Seb02]

2.3 Classification Techniques

Very different classifiers have been developed to meet the necessities of very different situations. In this section, some of the most known and widely used classifiers will be presented:

- *Naive Bayes Classifier*

  "The *Naive Bayes classifier* is perhaps the simplest and most widely used classifier." [APA+17, page 4]

  It is a probabilistic classifier that uses a probabilistic model to indicate if a document belongs to a specific category. Although it assumes that the distribution of the different words is independent of each other and "Even though this so called "naive Bayes" assumption is clearly false in many real world applications, naive Bayes performs surprisingly well." [APA+17, page 4]

  There are two main models associated to this classifier:

  1. *Multi-variate Bernoulli Model:* This model represents a document as a binary vector of features that indicates if a word is or is not present in the document. [APA+17]
  2. *Multinomial Model:* This model represents a document as a vector of frequencies of words. More information on this model can be found on [APA+17], but the main point is that this model always outperforms the Bernoulli model, except, sometimes, when the size of the vocabulary is small.

- *Nearest Neighbor Classifier*

  The nearest neighbor classifier uses distances to perform the classification. It is based on the idea that documents from the same category are similar. The classifier analyzes the document and will use a proximity function to find the documents that are more similar to him and assign the document to the category that has higher similarity. [APA+17]

  A special case of this classifier is the *k-nearest neighbor* which assigns the most common category of the k-nearest neighbors to the test document. [APA+17, MRS09]

- *Decision Tree Classifiers*

  Decision tree classifiers create a hierarchical tree of the training instances. This tree is called a decision tree, because it uses some attribute of the data to divide the data hierarchically, creating smaller subpartitions of the data as you descend along the levels. Like in any decision tree, each node of the tree tests a given condition of the test attribute reducing the size of samples that belong to the following nodes. [APA+17]

- *Neural Networks*

  Neural networks are composed by a network of units where the input units represent terms/-words and the output units represent the category or the set of categories. Internally, the different weights given to the different edges represent the relations of each term to the output category. One of the most used ways of training a document is through backpropagation where the weights of the edges might be changed, when an erroneous classification is made, to minimize the error. [Seb02]

- *Example-Based Classifiers*

  Example-based classifiers are also called lazy learners because they rely on labels that are attached to training documents instead of building a representation of the category. These classifiers find training documents that are similar to the test document and use the labels attached to the similar training documents to decide the category of the test document. [Seb02]

- *Support Vector Machines*

  Support Vector Machines (SVM) are supervised learning algorithms widely used for text classification. They are a form of *Linear Classifiers* which, in the case of text documents, perform a classification decision based on the value of the linear combinations of the features of the documents. Based on this, the output of a linear predictor can be defined as $y = \vec{a} \cdot \vec{x} + b$, where $\vec{x}$ is the vector with the normalized frequencies of the words of the document and $\vec{a}$ is the vector with the coefficients applicable to each corresponding frequency and b is a scalar. A single SVM can only separate two classes, one positive and one negative. A SVM algorithm tries to find a hyperplane with maximum distance d between the positive and negative examples. Those documents that have a distance d from the hyperplane are called *support vectors* and specify the actual location of the hyperplane. If the two classes are not linear separable, the algorithm tries to determine a hyperplane where the number of documents placed in the wrong class is minimal. [APA+17] To solve multiclass problems, several SVMs are usually used in which each one distinguishes between two classes [Bis06].

  One advantage of using this method is that it works really well with high dimensionality, the size of the feature space does not limit the usability of the method. It rarely needs feature selection since it selects data points (support vectors) required for the classification. [APA+17]

- *Ensembles*

  Classifier committees, most known as ensembles, are based on the concept that the combination of the classification of k experts may be better than the classification of only one. Transposing this concept to text categorization, the idea is to use k different classifiers and combine the outcome of each of them to obtain a decision. [Seb02]

  With that in mind, to use a classifier committee it is necessary to [Seb02]:

  - Choose k classifiers.

```
Input   : Document set D, similarity measure S, number k of clusters
Output  : Set of k clusters
initialization
Select randomly k data points as starting centroids.
while not converged do
Assign documents to the centroids based on the closest similarity.
Calculate the cluster centroids for all the clusters.
end
return k clusters
```

Listing 2.1: K-means clustering algorithm [APA+17]

- Choose/create a combination function.

Each classifier should be trained in an independent manner to potentiate the effectiveness of the classifier committee. [Seb02]

2.4 Clustering Techniques

Clustering is an unsupervised learning technique used when there is no labeled data, or there is interest in creating groups without relying on labels, to train a model capable of predicting a category. Below, a few types of clustering algorithms will be summarily described to give a sense of their similarity to the types of text classifiers referred above.

Hierarchical Clustering

Hierarchical clustering problems have this name because they create groups of clusters that can be seen as a hierarchy. The hierarchy can be constructed from top to bottom, a *divisive* approach, or from bottom to top, a *agglomerative* approach. They are *distance-based clustering algorithms*, which use a similarity function to measure the proximity of the documents. [APA+17, MRS09]

In the top-down approach, all the documents belong to the same cluster and then they are divided into sub-clusters. In the bottom-up approach, each document is considered an individual cluster and is then joined into other similar clusters until all documents belong to the same cluster. [APA+17]

K-means Clustering

K-means clustering divides documents into k clusters. The difficulty in this algorithm is on finding the optimal value of k to which value the algorithm is very sensitive. It is an NP-hard problem, which can be solved in a faster manner by efficient heuristics that converge rapidly to a local optimum.

The algorithm for k-means clustering can be observed on 2.1.

Topic Models

Topic modeling is a very popular probabilistic clustering algorithm to which we will refer in Chapter 3. The main point of topic modeling is to create a probabilistic model for the corpus of text documents. In this type of probabilistic clustering, documents are a mixture of topics, where each topic corresponds to a probability distribution of words. [APA+17]

A more detailed analysis of topic modeling should not be necessary for the overall comprehension of this thesis, but Allahyari (2017) [APA+17] is a good choice for it.

2.5 Conclusions

Text mining is a huge field of machine learning. Usually, documents are represented following the vector space model, which vector can be used to perform text categorization. As seen, there are several forms of text categorization: single-label classification, multilabel classification and ranking classification. Understanding the type of classification we want to perform is essential to decide which algorithms will be used.

In Chapter 3, it will be made a deeper analysis of the large range of algorithms that can be used for complaint analysis.

Chapter 3

State of the Art

The literature presents different ways of tackling the analysis of complaints. Throughout this chapter, the techniques and technologies used on some of the literature will be presented and analyzed, indicating the advantages, disadvantages and limitations of the approaches followed by the authors of each paper.

Although several works exist on analyzing user-generated content, they mostly study social media data [BT15], focusing on tasks such as sentiment analysis and opinion mining [PKF+13], or predicting the usefulness of product reviews [DN18]. Forte and Brazdil [FB16] focus on sentiment polarity of Portuguese comments, and use a lexicon-based approach enriched with domain specific terms, formulating specific rules for negation and amplifiers. Literature on (non-social media) complaint analysis is considerably more scarce, mainly due to the fact that such data is typically not publicly available.

3.1 Techniques Used for Complaint Analysis

In this section, an overview of the techniques that have been found will be performed by indicating which techniques have been used and in which situations. Because few literature was found on the topic and the information present in each paper can be considerably different from the others, the section will be divided into a few items, where each item starts with the title of the paper to which it refers and the conclusion will give an overview of the most promising approaches.

3.1.1 Linguistic-based approach

Ordenes et al. [OTB+14] propose a framework that provides a holistic approach for analyzing customer feedback by considering three key components of the value co-creation process: activities, resources, and context (ARC). To understand what is a co-creation process, it is easier to indicate that services consist of activities between customer and company in a co-creation process [OTB+14]. The framework goes beyond sentiment analysis and uses a linguistics-based text

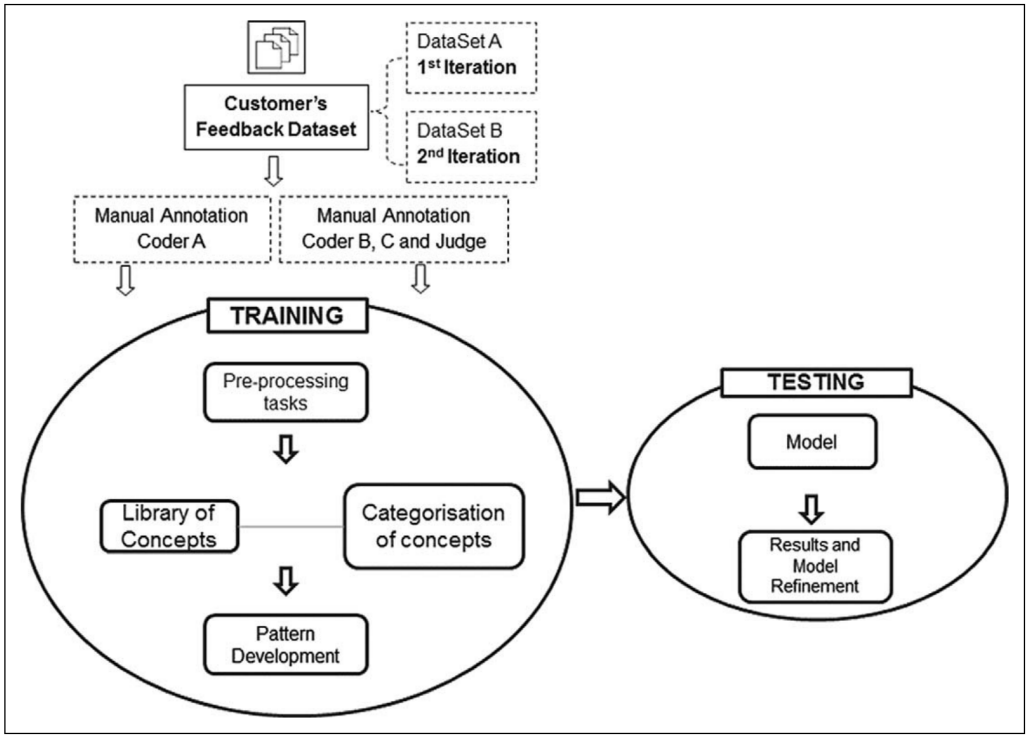

Figure 3.1: Text mining process used by Ordenes et al. [OTB+14]

mining model to automatically distinguish compliments from complaints, regarding different aspects of the customer feedback.

They followed the ensuing process [OTB+14]:

1. Business and data understanding.
2. Training or model development.
3. Import corpus (set of customer feedback documents gathered from the participant company).
   (a) Extract concepts using predefined built-in analyzers and dictionaries; evaluate and, if necessary, extend them. This stage is based on a manual coding process.
   (b) Define new concepts, patterns and text mining models; evaluate and, if necessary, extend them.
4. Testing or model evaluation.

As presented in Figure 3.1, the authors of the paper performed a first iteration where they used a model based on the manual annotation process of coder A. Then, they performed a second iteration where they used a model with new subcategories based on the manual annotation process of coders B, C and Judge and which employed sentence-level analysis through patterns. The new subcategories were created based on what they learned from the first iteration process.

The model performs the extraction of activities, resources and context (ARC) and the authors used the IBM Statistical Package for the Social Sciences Modeler to implement it.

The ARC framework possesses a few limitations identified by the authors [OTB+14, page 290]:

1. It is "specific to the situation and the type of service in which the research was employed."
2. It "requires work at the training stage to improve data capture and accuracy."
3. "Despite achieving a high level of accuracy (92%), data capture was low (just over 50%) in the first iteration."

The work focuses on a single activity domain, and in the end aims at obtaining a refined sentiment analysis model. In our case, we aim at distinguishing amongst a number of categories, without entering into a labor-intensive annotation process of domain-specific data.

3.1.2 Clustering-based approaches

Differently, Chugani et al. [CGR18] resorted to several different techniques to create clusters of complaints with the objective of allowing an interactive visualization of them. They refer the use of hierarchical clustering, k-clustering, multi-linear regression and outlier analysis as techniques and Jupyter and R as technologies.

This paper is not very detailed. The authors briefly explain how each technique works, but do not indicate how each one was implemented.

They indicate that, by using hierarchical clustering and k-means clustering, they got a better insight by having 5 clusters and that more clusters or fewer clusters had a lower density of most frequent words in comparison [CGR18].

In conclusion, the authors were able to show what problems customers were having, which is a valuable information for companies that want to improve their service [CGR18].

For this thesis, this paper has the limitation that it is based on clustering which is used for training with unlabeled data and the focus of this thesis is training with labeled data.

Kalyoncu et al. [KZYY18] approach customer complaint analysis from a topic modeling perspective, using techniques such as Latent Dirichlet Allocation (LDA) [BNJ03]. This work does not focus on automatically processing complaints, but instead on providing a visualization tool for mobile network operators.

The authors performed a visual analysis of topics of customer analysis complaints data which were inserted into their own proposed platform. They used the Gemsim library [ŘS10], which is built on top of NumPy[1], SciPy[2] and Pyro [BCJ+18], part of the open-source Python stack, to automatically extract semantic topics from documents using Latent Dirichlet Allocation. To extract useful information from the different topic models, they used LDAvis [SS14] which allows an interactive visualization.

[1] https://www.numpy.org
[2] https://www.scipy.org

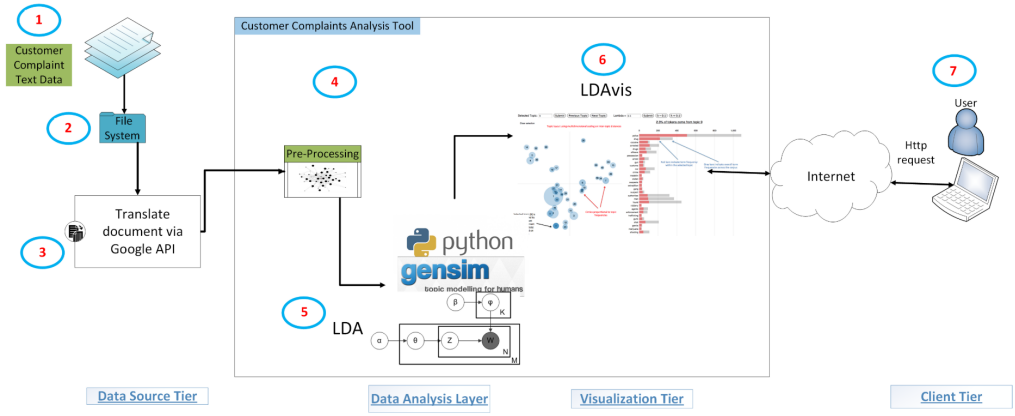

Figure 3.2: Setup used by Kalyoncu et al. (2018) [KZYY18]

The proposed tool is composed of four main modules: the Data Source Tier, the Data Analysis Tier, the Visualization Tier and the Client Tier [KZYY18].

As presented on Figure 3.2, the Data Source Tier obtains complaints from CSV files, translates the document from Turkish to English using Google Translator APIs[3] and sends the processed data to the Data Analysis Layer. The translation is made because the English language generates fewer features than the Turkish language [KZYY18].

The Data Analysis Layer performs the preprocessing stage, which removes newline characters, tokenizes words, removes stop-words, performs lemmatization and creates a dictionary and corpus required for the topic modeling procedure. When the preprocessing stage is completed, Latent Dirichlet Allocation is used for topic model generation. This step is done through the construction of a document-term matrix using the Gensim library [ŘS10].

Then, in the Visualization Tier, LDAvis is used for interactive topic model visualization [KZYY18].

To complete, the Client Tier interacts with the Visualization Tier to show the information to the user.

Like with Chugani et al. [CGR18], for this thesis, this paper has the limitation that it is based on clustering which is used for training with unlabeled data and the focus of this thesis is training with labeled data.

3.1.3 Single-label Supervised Classification

Dong and Wang [SZ15] aim to evaluate service quality applying text categorization on documents related to the processing of complaints by the call center staff. They start by preparing the data, dividing the complaints into those that met customer satisfaction and those that did not meet, transforming all data to UTF-8 encoding without byte order mark (BOM) and storing 80% of the documents to be used as training set and 20% to be used as testing set.

From now on, the process can be described using Figure 3.3.

[3]https://cloud.google.com/translate/

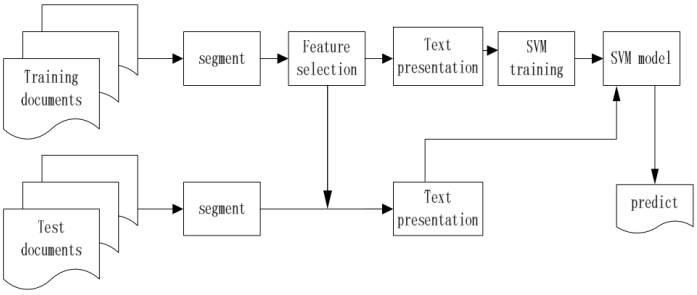

Figure 3.3: Text classification processes used by Dong et al. (2015) [SZ15]

After data preparation, the authors implemented word segmentation for training document sets and, then, extracted the feature vectors. Because some words are not present in general lexicon, the authors extended the dictionary function to allow the discovery of additional features [SZ15]. It is important to notice that the words to which we are referring can, in reality, be expressions with more than one word, but technically called "word" or "term" by the authors.

In the authors' case, 1420 feature words were extracted by word segment. This amount of features generates an extremely sparse vector space matrix that increases computational cost and reduces the ability of the classifier but also causes overfitting [SZ15]. To make the matrix denser, the authors adopted multiple filtration techniques [SZ15]:

1. Creation of a near-synonym dictionary and substitution of words with the same sense by the same one.

2. Use a term frequency value and eliminate terms whose term frequency value is below a given threshold.

3. Use Chi-square statistics and eliminate terms whose Chi-square value is below a given threshold.

After all filtration techniques mentioned above, the feature space matrix only contains values for the words that have meaning for the system.

Then, to have a better measure of the importance of the words in the collection of documents, the authors apply the calculation of the term weights through a formula that generates Term Frequency–Inverse Document Frequency (TF-IDF) weights [SZ15]. This phase corresponds to the phase "Text Presentation" in Figure 3.3, because, at this moment, it is possible to present the feature space matrix with TF-IDF values and easily understand which words have higher meaning for the system.

Moving to the next phase, the authors needed to decide which kernel they wanted to use because they are using SVM. They decided to go with Radial Basis Function (RBF) [Mit97] kernel which can handle the case when the relation between class labels and the attributes is nonlinear [SZ15]. With the kernel selected, it was necessary to identify which values should be given to the parameters C and gamma of the RBF kernel.

The authors performed a few experiments with different values of C and gamma and concluded that a better accuracy is obtained by reducing the number of dimensions of the feature vector. A table with the different values of C, gamma and number of dimensions and the corresponding accuracy scores can be consulted on the paper. The accuracy scores obtained demonstrate that using text categorization to evaluate the quality of the customer service can be an effective task. This is also the opinion of the authors [SZ15].

The text categorization system was built under Java platform, using several open source software, like IK Analyzer for word segmentation and LIBSVM [CL11] as classifier [SZ15].

3.1.4 Multilabel Supervised Classification

Fauzan and Khodra [FK14] propose the use of learning to rank to approach multilabel classification. To contextualize, the paper refers to LAPOR, Indonesia's government complaint management system.

According to the authors [FK14]:

> "LAPOR has unique structure consisting of two labels. First label is for the agency that is responsible to handle the complaint text, and second label is for some agencies that have connection to the complain text."

The paper proposes two different approaches to multilabel classification and learning to rank. The first one is to combine multilabel classification and learning to rank (Figure 3.4). The second one is to use learning to rank for multilabel classification (Figure 3.5).

The authors used a dataset provided by the government institution responsible to manage LAPOR. The dataset was relatively short, containing 2230 instances and 72 categories of agencies. Each instance is composed of 4 fields, including: the id, the complaint text, the topic and a label. According to the authors [FK14], each category has a different number of instances, making the dataset imbalanced.

The authors used only unique terms of complaint text as features and applied three types of weighting: binary (exists or not), term frequency and term frequency-inverse document frequency (TF-IDF). Additionally, they had two more features: the topic of the complaint text and the existence of the name of the agency in the text [FK14].

After selecting the features, the authors proceeded with the experiment by using 1230 instances as training set and the other 1000 instances as testing set. The experiment is divided into three scenarios that will be presented below.

Several different measurements and comparisons were performed by the authors using several classifiers.

Please note the following points for a better understanding of the conclusions that will be presented below:

- Single-label classification algorithms, multilabel classification algorithms and ranking classification algorithms were tested.

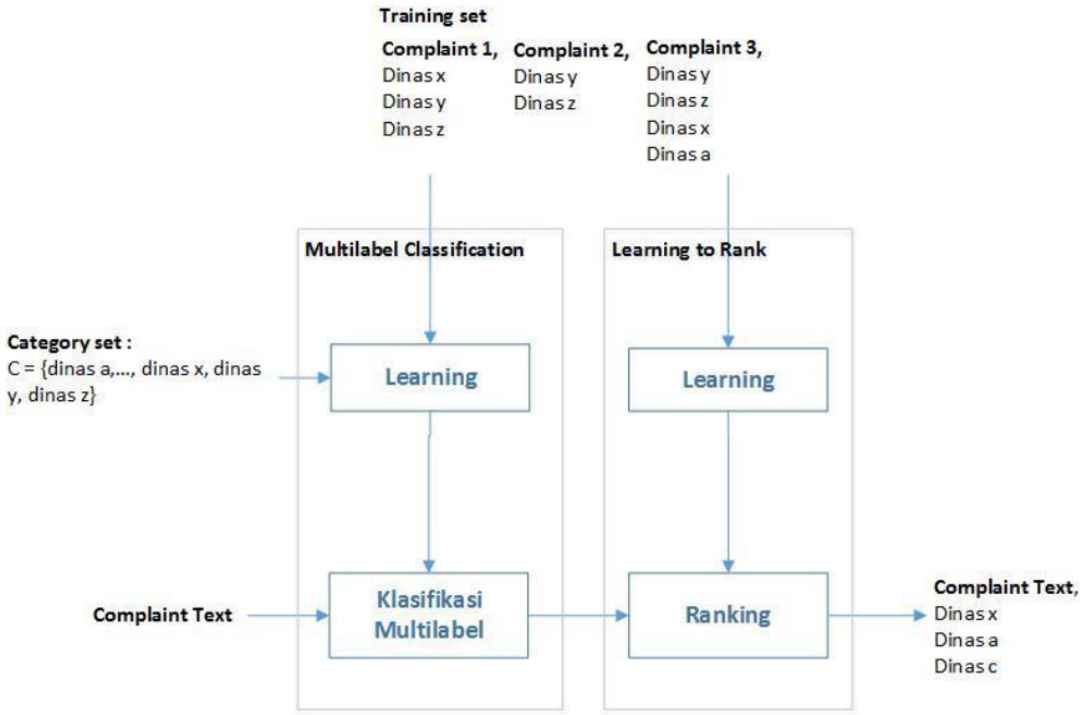

Figure 3.4: Process of combining multilabel classification and learning to rank proposed by Fauzan et al. (2014) [FK14] ("Dinas" means government agency.)

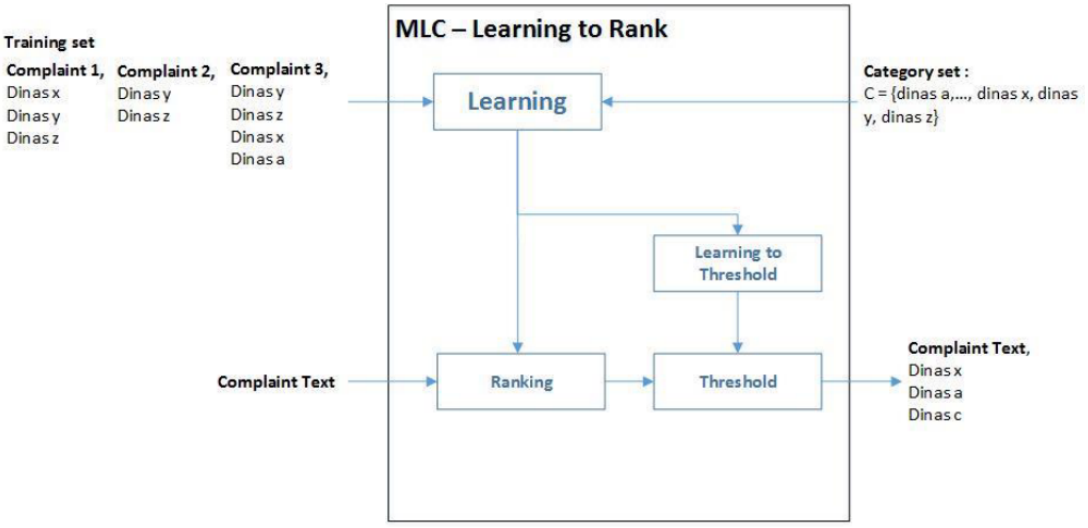

Figure 3.5: Process of using learning to rank for multilabel classification proposed by Fauzan et al. (2014) [FK14] ("Dinas" means government agency.)

- For single-label classification, the authors used Naive Bayes, J48 and SVM.
- For multilabel classification, the author used MLkNN, BR-Naive Bayes, BR-J48, BR-SVM, LP-kNN, LP-Naive Bayes, LP-SVM and LP-J48.
- For ranking classification, the authors used ListNet, MART, LambdaMART and Random Forests.
- Whenever single-label, multilabel or ranking classification is referred below, the authors performed tests with all the corresponding algorithms mentioned above.
- Each algorithm was run using binary weighting, term frequency weighting and TF-IDF weighting.
- Each algorithm was run using only the vector of terms, using the vector of terms and the complaint topic and using the vector of terms and the existence of label in the text.

In the first scenario, single-label and multilabel classification were used. The conclusion was that SVM is the best for single-label classification, except when using only the vector of terms. In this latter case, Naive Bayes outperforms SVM. The best performance of all for single-label classification was using term frequency and the vector of terms and complaint topic with SVM. For multilabel classification, the best performance was using ML-kNN with the vector of terms and existence of label in the text and using binary weighting.

In the second scenario, multilabel classification and ranking classification were used. The conclusion was that LP-SVM is the best for multilabel classification using the vector of terms and complaint topic with binary weighting. For ranking classification, all algorithms presented a great performance, but the best performance was using LP-SVM and Random Forests with the vector of terms and complaint topic and using binary weighting.

In the third scenario, only ranking classification was used. This scenario focuses on using learning to rank for multilabel classification. The conclusion was that LambdaMART with the vector of terms and complaint topic and using TF-IDF was the best choice [FK14].

After the experiments, the authors made a comparison of the accuracy of the different scenarios. The results show that the first scenario is the one with the best prediction for single-label classification, but the third scenario, i. e., using learning to rank, is always the better for multilabel classification. Otherwise, the second scenario is the one with worst performance. This bad performance is caused by the low accuracy of multilabel classification which affects learning to rank.

3.2 Available Resources

Several support technologies, widely used by the community to tackle NLP and ML related tasks, can be used for the analysis of complaints:

- *scikit-learn* [BLB+13] is a framework composed by several very efficient tools that are used for data mining and data analysis tasks. Specifically, it has tools oriented to work with text data. It is built on top of *NumPy*[4] and *SciPy*[5] which are also the base of other technologies that will be presented below.
- Natural Language Toolkit (NLTK)[6] is a platform for building Python programs to work with human language data. It provides lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing and others. There is also a book[7] freely available that explains how to use it.
- Keras[8] is an API for neural networks written in Python that can be run on top of TensorFlow and other platforms and can leverage CPU and GPU resources. It could be useful to use classifiers based on neural networks.
- TextBlob[9] is a Python library for processing textual data which provides a simple API for natural language processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation and more.
- StanfordNLP[10] [QDZM18] is a Python package for natural language analysis. It is composed by tools which can be used to convert free text into lists of sentences and words, to generate base forms of words and more. It has the advantage of being designed to be parallel among more than 70 languages, allowing an easy change of language and being capable of leveraging GPU resources on GPU-enabled machines.
- The IBM Statistical Package for the Social Sciences Modeler [OTB+14] is a paid software platform which offers statistical analysis, text analysis and a library of machine-learning algorithms.
- Gensim (LDA) [ŘS10, KZYY18] is a topic modeling tool designed for a scalable statistical analysis of the semantic structure of documents. It is built on top of NumPy, SciPy and Pyro. This tool is designed for clustering techniques.
- LIBSVM (SVM classifier) [CL11, SZ15] is an integrated software for support vector classification, regression and distribution estimation which supports multiclass classification. This tool supports different SVM formulations, various kernels and weighted SVM for unbalanced data.
- Weka[11] [FK14] is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules

[4] https://www.numpy.org
[5] https://www.scipy.org
[6] http://www.nltk.org/
[7] http://www.nltk.org/book/
[8] https://keras.io/
[9] https://textblob.readthedocs.io/en/dev/
[10] https://stanfordnlp.github.io/stanfordnlp/
[11] https://www.cs.waikato.ac.nz/ml/weka/

mining and visualization. There are also free online courses that teach machine learning and data mining using Weka.

- Mulan[12][FK14] is a Java library for learning from multilabel datasets. It contains algorithms classification, ranking and the combination of both. It also provides feature selection and classes that facilitate the evaluation and cross-validation.

- RankLib[13] [FK14] is a library containing learning to rank algorithms that supports MART, RankNet, RankBoost, AdaRank, Coordinate Ascent, LambdaMART, ListNet and Random Forests.

- Bidirectional Encoder Representations from Transformers (BERT) [DCLT18] is a method of pre-training language representations to generate word embeddings that can be used in NLP tasks.

3.3 Conclusions

There are several techniques and algorithms that can be used for complaint analysis. Those approaches that use clustering-based techniques appear to have low utility for this thesis because we aim to work with a dataset containing labeled data, but those techniques based on text classification are relevant for the task addressed in this thesis and promising results have been already reported in the literature. Based on the analysis of the papers presented on this chapter, especially the paper from Fauzan and Khodra [FK14], it appears that SVM is the best approach to follow when using single-label classification and that the best approach for multilabel classification is the use of learning to rank through LambdaMART.

Broadening the spectrum, traditional approaches to text categorization employ feature-based sparse models, recurring to bags-of-words and TF-IDF metrics and, only then, they apply other techniques presented above, like the use of synonyms [SZ15], LDA [KZYY18] or Chi-square statistics [SZ15] that reduce dimensionality, to obtain better models.

Dealing with complaints as a multilabel classification problem can be effective, even when the original problem is not, due to the noisy nature of user-generated content. Ranking algorithms [Li14, MCD15] are a promising approach in this regard, providing a set of predictions sorted by confidence. These techniques have been applied in complaint analysis [FK14], although with modest results.

On the technologies side, there are also several available resources that can be used to address the tasks we aim to tackle in this thesis. The ones that appear of higher value are scikit-learn, NLTK, TextBlob and StanfordNLP due to the amount of functionalities provided and due to being oriented to text analysis and natural language processing through the use of text classifiers.

[12] http://mulan.sourceforge.net/
[13] https://sourceforge.net/p/lemur/wiki/RankLib/

Chapter 4

Data Analysis

ASAE provided a dataset with several complaints and gave us a list of economic activities to which they can be assigned and the list of infractions that can be present in complaints. This chapter will explain the workflow used by ASAE to manage the complaints and will detail on the structure of the data provided and which adjustments to the data have been performed to pursue the tasks at hand.

4.1 ASAE workflow

Figure 4.1 presents a diagram of how ASAE officers manage each complaint. This workflow works as a basis for defining which tasks the prototype should perform to obtain an output similar to the one generated by an ASAE officer. Firstly, the prototype should receive a complaint and identify the economic entity to which it is targeted. Secondly, the prototype should obtain the economic activity present in the complaint accordingly to the list provided by ASAE. Thirdly, the prototype should identify which infractions are present in the complaint (if any) accordingly to the list given by ASAE. Finally, based on the infractions found, it is necessary to decide if the complaint is of ASAE competence and, if not, forward the complaint to the competent authority. If no infraction is present, the complaint can be archived, otherwise, if it is of ASAE competence, ASAE should proceed with an investigation.

4.2 ASAE dataset

The complaints dataset provided by ASAE comes in a structured format (Excel files), where several fields are structured data, but others, like the complaint text, are not, they are free text.

The dataset contains the textual content of the complaints together with metadata in one field and annotations, which are the outcome of its processing within ASAE, on the other fields. Some of the fields available are the following: entity with jurisdiction, state of the complaint, state of the

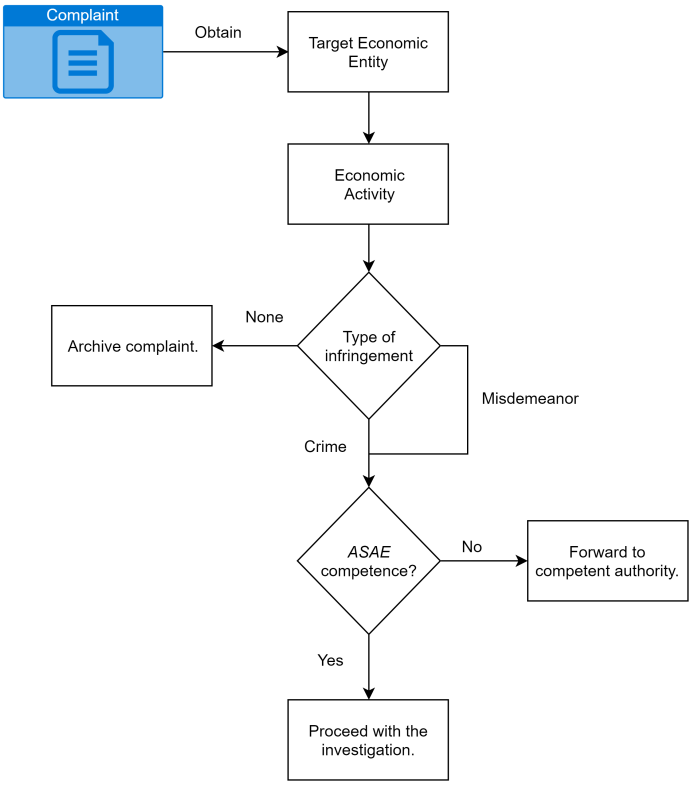

Figure 4.1: ASAE workflow

inquiry, archival date, how it was received (e-mail or form), sender type, the target economic agent and its data, the economic activities of the target economic agent, the types of infraction present ("crime", "misdemeanor" and a few others) and several other fields used for internal purposes.

From all the fields, three of them are very important for this thesis:

- The textual content of the complaint.
- The target economic activity.
- The evidence of infractions: misdemeanors or crimes.

Since the dataset already contains this information, labeled by an ASAE officer, it is possible to use it as a gold standard to train classifiers that will assign economic activities or infractions to new complaints.

Throughout the development of this thesis, 2 datasets were used based on the dataset provided by ASAE. The first dataset provided 48850 examples of complaints and the corresponding economic activities. This dataset is a subset of the second dataset with 79875 examples, filtered to contain only those complaints that are of ASAE competence. The second dataset provided 79875 examples of complaints, the corresponding activities and the most severe type of infraction present in each complaint.

The two datasets contain complaints received by ASAE between 2014 and 2018 and that were submitted by citizens, economic operators, public organizations or other organizations either by e-mail or through a contact form present in the official website[1].

Regarding the economic activities, there are 360 possible labels organized as a hierarchy. The first two levels are of no interest for the purpose of this thesis because they are administrative divisions. The third-level contains 4 classes and the fourth-level has 11 classes that will be presented later in this chapter. Each economic activity has a code corresponding to the fourth and lower levels of the tree. Each complaint might have more than one economic activity, but that is a rare case, which occurred about 200 times in 79875 complaints, so only one economic activity is considered for all complaints.

Regarding the infractions, there are 3103 possible labels. Similarly to economic activities, they are organized as a hierarchy. This hierarchy has a large amount of sub-classes, being extremely difficult to obtain classification models using a dataset with 79875 examples and each complaint might be associated with none or several infractions. To avoid the difficulty of handling a multilabel classification task, it was decided to reduce the scope of the analysis of the infractions to only discover the most severe type of infraction present (crime, misdemeanor or others), as will be explained in Section 4.2.2.

Due to the hierarchical nature of the labels, each class is composed of a number of sub-classes, which have a further decomposition level. For each sub-class happens that some of them have a large number of examples (thousands) and some reasonable cases have hundreds of examples, but the majority of them has a scarce amount of examples (dozens or even units). Given the large number of sub-classes, for the economic activities, it was decided to train the classifiers to predict fourth-level classes only, except on one experiment where the third-level is predicted before predicting the fourth-level.

Since our goal is to aid ASAE staff handling complaints, it was decided to base the classifications on their textual contents alone.

4.2.1 Economic activities

The economic activities taxonomy presents 11 fourth-level classes. The imbalanced distribution of these classes is shown in Table 4.1 and Table 4.2. Table 4.1 shows the distribution of the full dataset. Table 4.2 shows the distribution of the dataset with only ASAE competence complaints. The number of examples in both tables considers only the first economic activity present in the economic activities field because, as said earlier, it is not wanted to model the problem as multilabel due to the reduced number of learning instances in which there are multilabel annotations.

It is noticeable in Table 4.1 that class III has most of the examples (32.0%), followed by class IX (24.7%). The average complaint on this larger dataset is 1874 characters long, after removing HTML tags and other artifacts, containing information on its subject matter, the targeted economic agent and contact information of the claimant. In this dataset, there are only 2 examples for class

[1] http://www.asae.gov.pt/

Table 4.1: Distribution of economic activities (79875 examples)

3rd-level class	4th-level class	# examples	% examples
Food	I - Primary Production	294	0.368
	II - Industry	2343	2.933
	III - Restoration and beverages	25533	31.966
	IV - Wholesalers	355	0.444
	V - Retail	7517	9.411
	VI - Direct selling establishments	2	0.003
	VII - Distance selling (by Catalog and Internet)	3855	4.826
Production	VIII - Production and Trade	7586	9.497
	IX - Service Providers	19762	24.741
Safety	X - Safety and Environment	1220	1.527
Others	Z - No activity identified	11408	14.282

VI, being possible to perform the division into training and test subsets taking the distribution into account, so, although, this class could be removed due to the low number of examples, it was decided to include it in order to obtain classifiers that expect this type of data and check if the class is distinguishable from other classes with only one example.

Table 4.2: Distribution of economic activities (48850 examples)

3rd-level class	4th-level class	# examples	% examples
Food	I - Primary Production	134	0.274
	II - Industry	2031	4.158
	III - Restoration and beverages	20899	42.782
	IV - Wholesalers	299	0.612
	V - Retail	5951	12.182
	VI - Direct selling establishments	1	0.002
	VII - Distance selling (by Catalog and Internet)	2856	5.846
Production	VIII - Production and Trade	3335	6.827
	IX - Service Providers	9933	20.334
Safety	X - Safety and Environment	696	1.425
Others	Z - No activity identified	2715	5.558

Just like Table 4.1, in Table 4.2 it is noticeable that class III has most of the examples (42.8%), followed by class IX (20.3%); class VI has only one example. The average complaint on this smaller dataset is 1664 characters, long after removing HTML tags and other artifacts, containing information on its subject matter, the targeted economic agent and contact information of the claimant. When using this dataset, class VI is removed because it is not possible to divide the dataset in training and test subsets and take into account the distribution of examples with only one example of a class, two are the minimum required.

To complement the overview of the difference between Table 4.1 and Table 4.2, Figure 4.2 shows a bar chart highlighting the difference in distribution of each class. From the larger dataset to the smaller, classes I, IV, VI, IX and X present about half the quantity of complaints, having the same distribution, but the other classes have a larger representation. Class Z has a considerably smaller representation in the smaller dataset, indicating that a large part of the complaints belonging to this class are not of ASAE competence.

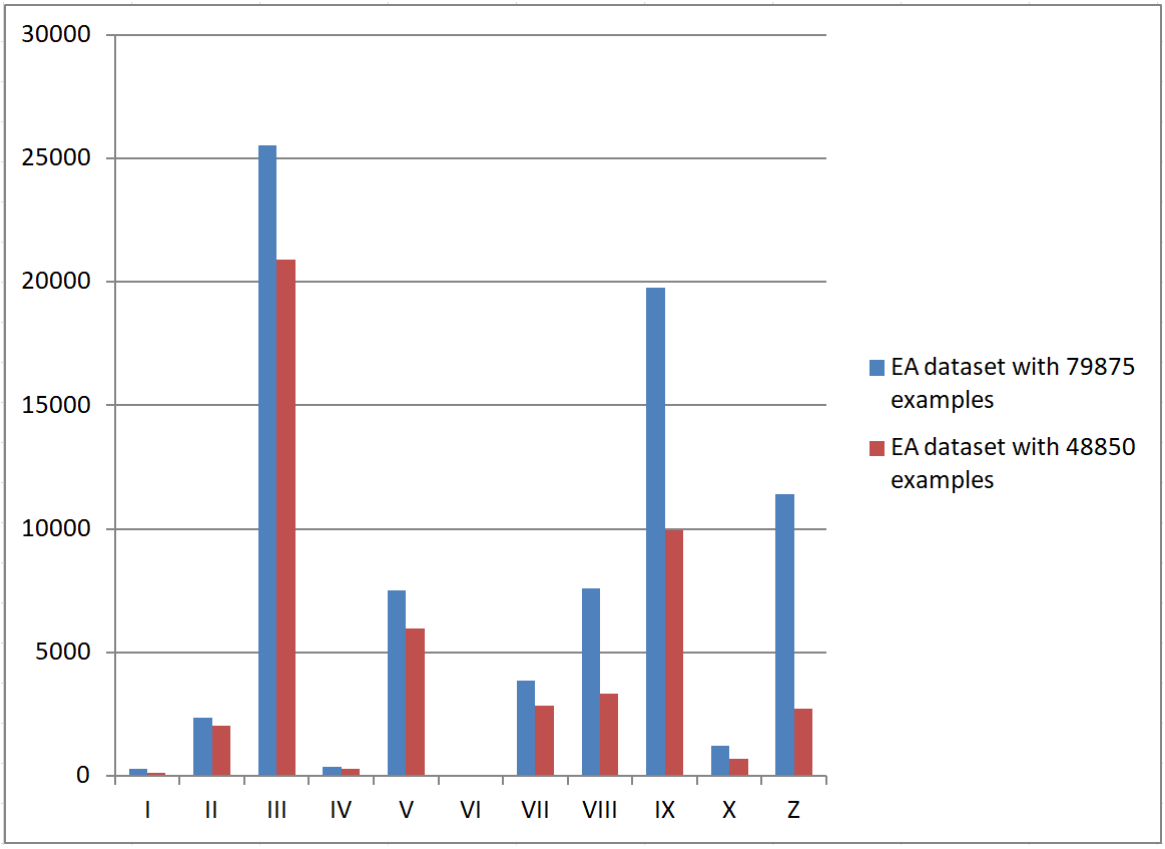

Figure 4.2: Large dataset of economic activities vs small dataset distribution

4.2.2 Infractions

The infractions taxonomy is extremely diffuse, so it was simplified to only analyze the types of infractions present in a complaint. Then, because ASAE officers have higher interest in knowing the most severe type of infractions, it was decided to account only for the most severe type of infraction present. The following three types of infractions are considered, sorted from most severe to less severe: “Crime”, “Misdemeanor” and “Other”. The most severe type of infraction present in a complaint corresponds to identify which types of infractions are present and choose the most severe.

Table 4.3: Distribution of higher infractions (79875 examples)

Class	# examples
Crime	4298
Misdemeanor	39896
Other	35681

Table 4.3 shows the number of examples for each type of infraction considering only the most severe infraction present in each example. For instance, if a given complaint includes both evidence of crime and misdemeanor, only the former is considered. This dataset is composed of 5.4% of crimes, 49.9% of misdemeanors and 44.7% of other situations where there is a minor

infraction or even none infraction exists.

There is no interest in presenting the distributions for the dataset that only contains examples of ASAE competence because it is necessary to know which possible infractions are present in a complaint to decide whether the complaint is of ASAE competence or not. Because this thesis focus on obtaining the most severe infraction instead of the exact infractions present in a complaint, it is not possible to use the classification to decide whether the complaint is of ASAE competence or not.

Chapter 5

Data Preprocessing and Feature Extraction

This chapter is dedicated to the data preprocessing and feature extraction techniques used along this work, the problems that were faced and the solutions found for them.

5.1 Preprocessing

The first step to generate a model is the preprocessing where the text is divided into simpler forms that can be used to extract features.

A typical text preprocessing pipeline was followed, using tokenization and lemmatization [MS99]. The tokenization is used to obtain all the words of the text divided and the lemmatization is used to reduce all inflected forms of a word to one single word, reducing the amount of features extracted and increasing the ability to identify features with high importance for the classification process.

NLTK [1], SyntaxNet [2], StanfordNLP [3] and spaCy [4] are some of the tools that can be used to perform these tasks [OT17]. However, since there is no conclusive comparison for Portuguese, a non-exhaustive experiment (included in Chapter 6, in Table 6.4) was performed to analyze which of the previously mentioned tools were more suitable to identify the economic activity of a complaint. StanfordNLP was chosen for most experiments because, as indicated in Table 6.4, the accuracy scores do not differ considerably and it provides several information that can be used to perform different experiments, giving a competitive contribution to the task. Specifically, it provides PoS tagging, which is explored to identify punctuation marks that are removed in all experiments and is used to perform some experiments. StanfordNLP also provides specific and complete support for Portuguese and several other languages and presents the data using the CoNLL-U

[1] https://www.nltk.org/
[2] https://opensource.google.com/projects/syntaxnet
[3] https://stanfordnlp.github.io/stanfordnlp/
[4] https://spacy.io/

format [QDZM18], which increases interoperability with other tools. StanfordNLP was used for preprocessing using a pipeline that starts with tokenization, then Part-of-Speech (PoS), followed by lemmatization and, finally, dependency parsing. Dependency parsing was used to be possible to use syntactic dependencies and other information to improve the selection of features without being necessary to preprocess the data again (a time-consuming task), but no syntactic dependency was found that could improve the classifier behavior, so its use became unnecessary.

After obtaining the lemmas, the prototype locates those words indicated as punctuation by the StanfordNLP and removes them. The identification is performed by looking at the UPOS field of each token and verifying if it is equal to "PUNCT". Stop words are words that are commonly used in different contexts and that do not add value to understand the content. Stop words are also removed after obtaining the lemmas by checking if each lemma occurs in the list of stop words. These stop words are obtained from the NLTK package of stop words that corresponds to the language in use and are present in Appendix A.

Before trying StanfordNLP, NLTK was used but, through the manual analysis of the lemmas obtained using it, it was considered that it is not very effective for Portuguese. NLTK was used with the MWETokenizer, which generates multi-word tokens, as tokenizer and, then, lemmas were extracted. Additionally, NLTK's stemming was also used before switching to StanfordNLP, but it generated expressions whose meaning could generate confusion and that did not match any stop word, so it was removed during the transition. One example is the following: the word is "moved", the generated lemma is "move" and the generated stem is "mov". If "move" was a stop word, it would be removed, but "mov" is not a word, so it could not be detected as a stop word.

When testing NLTK and spaCy as preprocessing tools, because NLTK does not recognize punctuation, the punctuation was removed comparing the lemmas obtained with the punctuation list provided by Python's "string" package. spaCy was used as tokenizer and lemmatizer recurring to two different models: "pt_core_news_sm" and "xx_ent_wiki_sm".

Finally, the removal of adjectives was tested by removing tokens identified by StanfordNLP as adjectives, and the use of near-synonyms was tested using CONTO.PT [Gon16]. These two tasks were employed as experiments because they reduce the amount of features extracted from the text. Further details on the idea behind each experiment will be detailed in Chapter 6. Although they are preprocessing steps, they are performed during the preparation for the extraction of features so that it is not necessary to preprocess all the data again (a time-consuming task). NLTK and spaCy are two of the libraries that provide ways of obtaining near-synonyms. As these two tools, most of the datasets found for the identification of near-synonyms only worked for English, using, in general, thesaurus.com, confirming that the Portuguese language has limited resources for NLP.

On the dataset with 48850 examples, there are 13 examples that are ignored because they cannot be preprocessed and 1 example which is ignored because it is the only example for class VI. The explanation for the removal is in Chapter 4. On the dataset with 79875 examples, there are 19 examples that are ignored just because they cannot be preprocessed. Usually, examples that cannot be preprocessed are examples that do not contain valid text, like empty examples, but some of the examples cannot be preprocessed because the CoNLL-U information apparently does not

follow the standard, not being completely valid.

StanfordNLP provides several more information than NLTK, but it is based on deep learning techniques, requiring a lot more processing power and time, so it was necessary to apply some procedures to use it:

- Store the preprocessed data to load it later.
- Allow to stop and resume the preprocessing.
- Use Python generators to load the data.

5.1.1 Preprocessed Data Storage

As indicated above, it was necessary to store all the data generated during the preprocessing phase to load it again later. This was necessary because the amount of data generated was so large that could occupy all RAM and kill the program and because the preprocessing was too slow. Initially, the data was stored using the pickle protocol, but the aim was to be able to analyze the generated data in a text format, so the prototype was adjusted to store the data as JSON using the "jsonpickler" package to keep the class structure. Quickly it was found that the load time was too slow, so it was necessary to find another way of storing the data.

Table 5.1: Time and space performance of different storage methods

Method	Time (ms)	Space 1 (bytes)	Space 2 (bytes)
jsonpickle.dumps()	1229.3346	58	1491
jsonpickle.loads()	2504.6164	-	-
pickle.dumps()	52.0222	47	1163
pickle.loads()	45.4727	-	-
_pickle.dumps()	51.3202	47	1163
_pickle.loads()	43.4479	-	-
ujson.dumps()	43.5423	56	Error
ujson.loads()	27.4412	-	-
json.dumps()	289.0085	58	Error
json.loads()	266.1309	-	-
marshal.dumps()	66.5040	53	1002
marshal.loads()	27.4707	-	-

Table 5.1 presents the results obtained on an experiment conducted to decide which was the best technique to store the data and load it in a fast manner. The second column indicates the time spent by the corresponding method to store 100 thousand times the Python list "[1,2,3]". The third column indicates the space occupied by the dumped version of the Python list "[1,2,3]". Finally, the fourth column indicates the space occupied by the dumped version of the dictionary of an example of the Document class, which is part of our code, only with the initial data, the data generated after the preprocessing is not considered here. The Document class is a class that stores all the data related to one example. For each line of the Excel file provided to the prototype, a Document object is created which contains the number of the row on the Excel file, all the different fields with data of the row in analysis and, after preprocessing, it will contain the data

generated during preprocessing, i. e., tokens, lemmas, PoS information and all other information generated by StanfordNLP.

The experiment was conducted after detecting that it would be necessary a few hours for the completion of the load using "jsonpickle" when it was necessary that the load occurred in a maximum of a few minutes.

As demonstrated in the table, the use of "jsonpickle" is too slow, specially during loading. "pickle" is the fastest option that supports our Document class, also having a good performance in terms of space usage.

After solving this situation related to the loading time, another problem needed to be solved. The size of the data put in RAM was of a few gigabytes because the prototype was dumping and loading all documents at once to a file. A second try, was to dump and load each document to a file, creating as many files as documents. This was too slow due to the number of calls to the disk. A third try passed by creating files of several documents, but each file having about 100MB. This was a very good approach, but the size of the documents in RAM was larger than the 100MB in disk. Finally, it was found that Python uses generators, allowing to easily and efficiently load document by document from a single file, which was the best option with an unnoticeable footprint on RAM.

Using generators, it was also tried to use multiprocessing and multithreading to load documents while the tasks performed to each document were being done, but it slowed the loading process, so only one process/thread is being used.

5.2 Feature Extraction

Before performing the extraction of features, it is necessary to remove classes containing only 1 example in the whole dataset (class VI in Table 4.2). It is important because the classifiers receive a training set that is obtained taking into account the distribution of classes (stratified split). The stratified split only requires that any class has at least two examples. In the dataset with 48850 examples, the class VI is the only one that presents only 1 example, being necessary to remove it, but, in the dataset with 79875 examples, the class VI presents 2 examples, not being necessary to remove it. Because the amount of examples is so small, the examples that are related to this class could be removed, but that is not done, as explained in Chapter 4.

To perform the extraction of features, different techniques were experimented. Count, Hashing and TF-IDF are provided by scikit-learn [5], an API that allows to perform several of the steps necessary for different machine learning tasks [BLB+13]. Bidirectional Encoder Representations from Transformers (BERT) [DCLT18] is provided by flair [6]. The count technique transforms a collection of texts into a matrix of token counts. The hashing technique also transforms a collection of texts into a matrix of token counts or binary occurrences, depending if we want counts or if want to obtain one-hot encoding. It was used to obtain token counts and compare the difference with the

[5] https://scikit-learn.org/stable/

[6] https://github.com/zalandoresearch/flair

count technique. It has a few advantages, like low memory scalability, but differs from the count technique because there is a few risk of collisions. The use of the hashing technique was also motivated by situations where the use of the count technique with SVM was not able to converge in acceptable time. The TF-IDF technique transforms a collection of texts into a matrix of TF-IDF features which represent the importance of each token in the collection of all documents. Finally, BERT is based on embeddings and was created by Google. It was tested because people related to the IA.SAE project used it and obtained interesting results which are confirmed by Akbik et al. [ABV18]. It was used recurring to the model "bert-base-multilingual-uncased". Because it is limited to 512 tokens by example given to embed, the prototype performs the division of the example in smaller parts and gives the smaller parts to be embedded by BERT. In the end, the average of the embeddings obtained from the smaller parts is placed in the features matrix. BERT showed the same problem as the count technique. When used with SVM with linear kernel, SVM cannot converge in acceptable time without performing adjustments to the parameters of SVM. These adjustments will be indicated in the corresponding experiences.

1-grams TF-IDF, which provides an overview of the importance of a given token in the whole set of texts/documents, was mainly used because it presented the best results both in the related work analyzed as in our experiments which were performed with the different data representation techniques presented above and will be presented in Chapter 6 and Chapter 7.

For the count, hashing and TF-IDF techniques, it is possible to use different quantities of n-grams. The results obtained by using bags-of-words of 1-grams, 2-grams, 3-grams and intervals of 1 to 2-grams, 1 to 3-grams and 2 to 3-grams will be presented on the Chapters 6 and 7.

Feature selection techniques, which filter some of the features generated by TF-IDF, were also tested. The experiments will be presented on the Chapters 6 and 7.

Chapter 6

Economic Activity Prediction

Based on the workflow present in Figure 4.1, the first task that was addressed was the retrieval of the economic activity associated to a complaint. To discover the best techniques that should be used to obtain a higher accuracy score in the retrieval of economic activities, several different experiments were performed and will be presented throughout this chapter.

All classification tasks presented were performed using a training subset composed of 70% of the dataset and a test subset composed of 30% of the dataset. For the dataset with 79875 examples, the training and test subsets were always the same, except when the classes changed, i.e., when the number of classes was different than 11. For the dataset with 48850 examples, the training and test subsets were always regenerated before executing the pipeline of classifiers, reducing slightly the confidence which we can give to them. In both cases, a stratified split was performed, i.e., every time the training and test subsets were generated, the division took into account the proportion of examples of each class in both subsets. More concretely, the stratified split ensures that the distribution of examples of a given class in the training set is similar to the distribution in the test set, ensuring that the trained classifier learns the real distribution of the data. The stratified split must be ensured for the following reasons:

- ML using statistical techniques captures the distributions of the classes during training, being important to test under the same conditions.
- If the reality of the data is a given distribution, then, if the data is divided into parts, it is necessary to keep that distribution in the partitions. Otherwise, a realistic representation of the data is not being used.

For the classification task addressed in this chapter, it was decided to only consider the fourth-level of the hierarchy on most experiments. As mentioned in Chapter 4, most of this level's subclasses do not have enough representation to allow the classifiers to distinguish them.

In order to find out which classifiers would allow to obtain the best results, it was decided to use Random Forests, Bernoulli NB, Multinomial NB, Complement NB, k-Nearest Neighbors,

SVM, Decision Tree, Extra Tree and Random (stratified). This was done for the first experiments using the dataset with 48850 examples. When using 79875 examples, only Random Forests and SVM with linear kernel are used because they are the ones with more promising results. Some experiments that were performed using the dataset with 48850 examples were not repeated for the dataset with 79875 examples due to time constraints. All of them were implemented using scikit-learn[1] and the default parameters (for version 0.22) were used, except when the experiment required that they were changed. In those cases, it will be explicitly stated while describing the experiment.

The accuracy score was used as main performance metric, instead of the average macro-F1 score, because the aim is to provide a list of classes sorted by confidence and it is not critical to correctly classify minority classes. The macro-F1 score is presented in several experiments for scientific reasons and completeness.

Throughout the chapter will be presented several tables whose nomenclature is as follows:

- Acc@k: accuracy scores considering the top-k predicted classes, according to the confidence of the classifier predictions.
- Macro-F1@k: average macro-F1 scores considering the top-k predicted classes, according to the confidence of the classifier predictions.

6.1 Baseline

To obtain a baseline, a configuration based on what appeared to obtain the best results based on the related work analyzed was used. The experiment was performed using StanfordNLP in preprocessing and resorting to bag-of-words of 1-grams and TF-IDF encoding (in short: 1-grams TF-IDF) to obtain the data representation. The experiment was performed both in the dataset with 48850 examples and in the dataset with 79875 examples.

Taking into account the potential usage of the classifier, which is meant to help humans on analyzing complaints by providing likely classification labels (as opposed to imposing a definitive one), we looked at the performance of the classifier considering the ranking provided.

The ranking is computed by obtaining the list of classes that the classifier received during training and the list of probabilities generated by the classifier for each example during prediction. The index of each probability in the probabilities list is the same as the index of the class in the classes list, so it is only necessary to create a mapping and sort the classes from the highest probability to the lowest.

For the dataset with 79875 examples, one can see in Table 6.1 that Random Forests is not the best option because it only correctly predicts 61.96% of the examples provided. However, as we are interested in providing a sorted set of options, it becomes reasonable because the accuracy improves substantially when analyzing the top-2 and top-3. In terms of macro-F1, the values obtained are very limited, due mainly to the fact that the dataset is very unbalanced, which leads

[1] https://scikit-learn.org/stable/

Table 6.1: Economic Activity Multiclass Classification scores for top-k predictions (79875 examples)

Classifier	Acc@1	Acc@2	Acc@3	Macro-F1@1	Macro-F1@2	Macro-F1@3
Random Forests	0.6196	0.7840	0.8817	0.36	0.5	0.59
SVM (linear)	**0.7435**	**0.8863**	**0.9451**	**0.57**	**0.71**	**0.77**

to the classes with the greatest number of elements being predicted more times. SVM with the linear kernel has obtained quite good results, as it accurately predicts 74.35% of the examples provided and the precision increases considerably when analyzing the top-2 and top-3, achieving 94.51% of correctly predicted examples which is a great help for a human operator. In terms of macro-F1, the values obtained using SVM are reasonable and probably are not better because the dataset is unbalanced.

Table 6.2: Confusion matrix of the baseline SVM (Top-1) (79875 examples)

		Predicted									
		I	II	III	IV	IX	V	VI	VII	VIII	X	Z
Actual	I	46	4	3	0	0	6	0	0	2	0	27
	II	2	368	121	1	38	61	0	0	10	0	101
	III	5	73	6917	6	248	83	0	5	32	5	284
	IV	0	9	6	31	10	19	0	1	2	0	29
	IX	0	10	353	7	4763	33	0	70	165	11	516
	V	3	29	167	9	51	1729	0	13	51	1	202
	VI	0	0	0	0	0	0	0	0	1	0	0
	VII	0	0	6	0	78	6	0	892	41	0	133
	VIII	1	10	115	12	250	55	0	47	1367	4	414
	X	0	6	11	2	88	10	0	8	67	104	70
	Z	13	50	390	15	801	154	0	107	282	15	1594

Analyzing the confusion matrix in Table 6.2 that was obtained for SVM accepting the top-1, it is concluded that most of the examples of the different classes are correctly classified, supporting the accuracy score obtained. Class III has several examples, it is the majority class, being erroneously predicted a few times and class VI is never predicted, indicating that adding it to the training subset does not reduces considerably the accuracy score.

Table 6.3: Economic Activity Multiclass Classification accuracy scores for top-k predictions (48850 examples)

Classifier	Acc@1	Acc@2	Acc@3
Random Forests	0.6787	0.8322	0.8885
Bernoulli NB	0.5115	0.7533	0.7913
Multinomial NB	0.4603	0.6790	0.7936
Complement NB	0.5914	0.8214	0.8873
K-Neighbors	0.6283	0.7699	0.8447
SVM (linear)	**0.8075**	**0.9031**	**0.9474**
NuSVM	0.7995	-	-
Decision Tree	0.6659	0.7086	0.7226
Extra Tree	0.5056	0.5627	0.5703
Random - stratified	0.2504	0.3081	0.3241
SGD	0.7870	-	-
Bagging	0.7254	-	-

Regarding the dataset with 48850 examples, Table 6.3 presents the accuracy scores obtained using several different classifiers. SVM (with the linear kernel) is again considered the best one and it must be because it handles well the imbalance in the quantity of examples for each class present in the dataset.

Although NuSVM obtained a very good accuracy, it was tested only for this experiment because it takes a long time to train and specially because the value for the 'nu' parameter depends on the dataset. In this experiment, different maximum values for 'nu' were found depending on the size of the dataset. Because it is expected that the prototype is able to be improved along time, the use of an algorithm whose parameters need to be redefined is not suitable.

6.2 Preprocessing

6.2.1 Preprocessing tools

In order to decide the best preprocessing tool to be used, an experiment was carried out comparing 3 of the tools with the best results in the related work analyzed. From the tools analyzed by Omran et al. [OT17], NLTK, StanfordNLP and spaCy were tested.

For this experiment, 1-grams TF-IDF was used to obtain the data representation and the top-1 accuracy score is taken into account. This experiment was only performed for the dataset with 48850 examples, as it was performed at an early stage of the study and the results were fairly similar. Considering that the distribution of classes does not change much between datasets, a big difference in the results is not expected.

Table 6.4: Economic Activity Multiclass Classification accuracy scores using different preprocessing tools (48850 examples)

Classifier	StanfordNLP (baseline)	NLTK	spaCy - pt_core_news_sm	spaCy - xx_ent_wiki_sm
Random Forests	0.6787	0.6924	0.6818	0.6911
Bernoulli NB	0.5115	0.5363	0.5110	0.5185
Multinomial NB	0.4603	0.4719	0.4613	0.4648
Complement NB	0.5914	0.6263	0.5965	0.6066
K-Neighbors	0.6283	0.3146	0.6328	0.6180
SVM (linear)	**0.8075**	**0.8164**	**0.8093**	**0.8135**
Decision Tree	0.6659	0.6698	0.6669	0.6703
Extra Tree	0.5056	0.5228	0.5185	0.5166
Random - stratified	0.2504	0.2540	0.2510	0.2499

Table 6.4 presents the accuracy scores obtained using different preprocessing tools to perform tokenization and lemmatization. NLTK is the tool that obtained the best overall accuracy score, followed by spaCy and, in last, is StanfordNLP. This was surprising because the non-exhaustive manual analysis of the lemmas obtained by NLTK performed before this experiment indicated that some lemmas were not correct. The explanation found for this situation is that although the lemmas might not be the correct ones they have a one-to-one correspondence to more original tokens that results in better features.

StanfordNLP was chosen for further use because the performance loss is negligible, it provides PoS information and dependency parsing, and the accuracy scores of the different tools do not differ much. More specifically, StanfordNLP indicates if a given lemma is punctuation or an adjective, feature that we use to remove punctuation on all experiments and that we used in an experiment where we test the removal of adjectives. Additionally, StanfordNLP provides specific and complete support for Portuguese and several other languages and presents the data using the CoNLL-U format [QDZM18] which can be used as a standard, allowing its use with any tool that supports CoNLL-U.

6.2.2 Impact of HTML

Although the dataset provided already was preprocessed to remove HTML, it was possible to find occurrences of it in some examples. Because the HTML has no meaning for the classification task, it was decided to remove it again and analyze the influence of the removal in the classification.

Table 6.5: Economic Activity Multiclass Classification accuracy scores removing HTML (48850 examples)

Classifier	No removal (baseline)	Words only	No symbols	StanfordNLP with removal
Random Forests	0.6787	-	-	0.6782
Bernoulli NB	0.5115	0.4882	0.4843	0.5144
Multinomial NB	0.4603	0.4577	0.4540	0.4626
Complement NB	0.5914	0.5751	0.5571	0.5957
K-Neighbors	0.6283	0.2519	0.2512	0.6458
MLP	-	0.4893[2]	-	-
SVM (linear)	**0.8075**	**0.8053**	**0.8024**	**0.8128**
Decision Tree	0.6659	0.6430	0.6455	0.6742
Extra Tree	0.5056	0.5036	0.4955	0.5148
Random - stratified	0.2504	0.2534	0.2551	0.2567
SGD	0.7870	-	-	0.7881

To remove HTML and preprocess using StanfordNLP takes several days, so, for first approach, it was performed a tokenization by spaces, followed by the removal of all characters that were not letters in each token and ignoring empty tokens. The idea behind this tokenization was to obtain only words, removing all numeric and symbolic characters. The results using 1-grams TF-IDF can be seen in column "Words only" in Table 6.5. In all cases, the results are below the baseline, but, interestingly, the difference for SVM is of 0.22% indicating that other data has low value and that a simpler preprocessing process could be used. Only K-Neighbors has an important improvement with the use of StanfordNLP. Additionally, only for this case, a Multi-Layer Perceptron (MLP) was tested obtaining 48.93% of accuracy, but it was interrupted due to the amount of time it takes to converge, so this result is not interesting and cannot be compared.

Then, a second approach was considered by performing a tokenization by spaces, followed by the removal of all characters that were not letters nor numbers in each token and ignoring empty tokens. The concept behind this tokenization was to remove every symbolic character. The results

[2]Training interrupted

using 1-grams TF-IDF can be seen in column "No symbols" in Table 6.5. In most cases, the results are below the baseline and below the "Words only" version, indicating that numbers in tokens confuse the classifier.

Finally, the usual pipeline of using StanfordNLP and TF-IDF with 1-grams, but removing HTML, was approached to be completely sure that the HTML was not lowering the scores. The results can be seen in column "StanfordNLP with removal" in Table 6.5. In all cases, the removal obtained a slight increase in the accuracy of the classifiers. This indicates that the presence of HTML limits the performance of the classifiers, being necessary a better removal.

Because HTML was removed during the creation of this smaller dataset and because the difference in accuracy is small, it was decided that the removal of HTML is not necessary for the further experiments and the personnel that provided the dataset was informed that the HTML removal tool used was keeping some HTML.

6.3 Feature extraction

6.3.1 Data representation techniques

From the different possible data representation techniques, four were tested: Count, Hashing, TF-IDF and BERT.

Table 6.6: Economic Activity Multiclass Classification scores using different data representation techniques (79875 examples)

Classifier	Count	Hashing	TF-IDF (baseline)	BERT
Random Forests (Acc@1)	**0.6359**	0.6138	0.6196	0.4839
Random Forests (Acc@2)	**0.7938**	0.7751	0.7840	0.6739
Random Forests (Acc@3)	**0.8856**	0.8776	0.8817	0.7971
SVM (linear) (Acc@1)	0.4503	0.7106	**0.7435**	-
SVM (linear) (Acc@2)	0.6417	0.8586	**0.8863**	-
SVM (linear) (Acc@3)	0.7756	0.9297	**0.9451**	-
Random Forests (Macro-F1@1)	**0.38**	0.36	0.36	0.22
Random Forests (Macro-F1@2)	**0.52**	0.5	0.5	0.35
Random Forests (Macro-F1@3)	**0.60**	0.59	0.59	0.44
SVM (linear) (Macro-F1@1)	0.31	0.52	**0.57**	-
SVM (linear) (Macro-F1@2)	0.45	0.66	**0.71**	-
SVM (linear) (Macro-F1@3)	0.56	0.73	**0.77**	-

The results obtained by using 1-grams Count, Hashing and TF-IDF on the dataset with 79875 examples are present in Table 6.6 and indicate that the count technique is better for Random Forests and TF-IDF is better for SVM. BERT obtains the worst results for all tested cases. The count technique was tested with SVM limited to a maximum of 1000 iterations due to the amount of time necessary to converge and, like in Table 6.7, it could not converge. The hashing technique was tested to compare with the count technique because both count the number of occurrences, occurring that hashing uses an hash instead of the feature name. Its use was motivated by the difficulty in convergence when using the count technique with SVM. Several tries were performed to use BERT, but due to the amount of time it takes to run and due to problems of resource

starvation when the execution was almost finishing, the results for its use with SVM could not be obtained.

Table 6.7: Economic Activity Multiclass Classification accuracy scores using different data representation techniques (48850 examples)

Classifier	Count	Hashing	TF-IDF (baseline)	BERT
Random Forests	0.6958	0.6561	0.6787	0.5473
Bernoulli NB	0.5115	0.4415	0.5115	0.3162
Multinomial NB	0.6329	Error[1]	0.4603	Error[1]
Complement NB	0.6790	Error[1]	0.5914	Error[1]
K-Neighbors	0.5359	0.5750	0.6283	0.5121
SVM (linear)	**0.7784**[2]	**0.7953**	**0.8075**	**0.6783**
Decision Tree	0.6786	0.6671	0.6659	0.3908
Extra Tree	0.4968	0.4865	0.5056	0.3518
Random - stratified	0.2442	0.2509	0.2504	0.2553
SGD	0.6320	0.7374	0.7870	0.5903
Bagging	0.7317	0.7290	0.7254	0.5140

[1] Hashing and BERT may generate negative feature values, not supported by some classifiers.
[2] Failed to converge after 1000 iterations.

Table 6.7 presents the accuracy scores obtained using the different data representation techniques presented in Section 5.2. The accuracy scores obtained are very variable depending on the classifier used. Because we are focusing on Random Forests and SVM (with linear kernel), considering that they should provide the best accuracy scores, we can conclude that for the first the count technique is the best, but for SVM the TF-IDF technique is the better. It is important to note that SVM using the count technique was not able to converge even increasing the number of iterations, so its accuracy could be higher with a larger number of iterations. Regarding BERT, the low performance might be related to the fact that it is based on embeddings and the model used, a multilingual model, might not contain features corresponding to all the lemmas presented, losing features that the others do not lose.

To use the count technique in conjunction with SVM it is necessary to perform a few adjustments in the SVM parameters so that SVM can converge in an acceptable amount of time. One way is to disable the use of probabilities and SVM will only give the class it considers the correct, not being possible to obtain a ranking, but this decreases considerably the execution time. The other way is to limit the number of iterations. In Table 6.7, when using the count technique, SVM was limited to 1000 iterations, failing to converge until this limit was passed.

Because the best accuracy score is the one obtained using SVM using TF-IDF encoding, the following experiments will use TF-IDF to create a representation of the features extracted.

In general, classifiers were trained using TF-IDF to extract features, being important to note that TF-IDF extracted 252000 features, but, only 101159 have non-null importance in Random Forests, which demonstrates that, although a lot of features are extracted, many of them (about 60%) have no utility for the classifier.

6.3.2 Impact of n-grams

The generation of features with different amounts of n-grams can be of interest for the classification task, allowing to disambiguate some situations like negations.

Table 6.8: Economic Activity Multiclass Classification accuracy scores using different n-grams (48850 examples)

Classifier	1-gram	1 to 2-grams	2-grams	1 to 3-grams	2 to 3-grams	3-grams
Random Forests	0.6737	0.6503	0.6323	0.6230	0.6127	0.5663
Bernoulli NB	0.5115	0.4763	0.4703	0.4622	0.4561	0.4495
Multinomial NB	0.4603	0.4568	0.4700	0.4568	0.4683	0.4733
Complement NB	0.5914	0.5432	0.5978	0.5381	0.5922	0.6320
K-Neighbors	0.6283	0.6152	0.5821	0.5950	0.5631	0.5413
SVM (linear)	**0.8075**	**0.8098**	**0.7640**	**0.8004**	**0.7396**	**0.6532**
Decision Tree	0.6659	0.6729	0.6121	0.6717	0.6120	0.5413
Extra Tree	0.5056	0.5338	0.5462	0.5541	0.5398	0.5298

Table 6.8 presents the accuracy scores obtained using different n-grams when performing feature extraction with TF-IDF and accepting only the top-1 predictions. By observing the results, it is possible to conclude that the interval of n-grams to choose depends on the classifier.

For SVM, 1 to 2-grams is the best choice, followed by 1-gram. This difference in accuracy score might be related to cases where quantification is used (e.g. “never occurred”).

For SVM, 1 to 2-grams is slightly better than 1-grams, but, for Random Forests, 1-grams is considerably better than 1 to 2-grams. Due to this reason, the following experiments use only 1-grams.

6.4 Feature selection

6.4.1 Feature selection techniques

Feature selection through the use of Latent Dirichlet Allocation (LDA) was tested because, as said in the beginning of the chapter, it was detected that a large quantity of features extracted had a null importance for Random Forests, so it is expected that LDA could allow to obtain higher accuracy scores by grouping features into sets that are semantically related that can be more efficiently used to train the classifiers.

As it is possible to note by observing Table 6.9, the use of LDA reduced largely the effectiveness of the classifiers and, although Random Forests presents an increase of 6% that could indicate that more topics would allow better filtering, almost all other classifiers show that the increase in the number of topics maintains or even decreases the accuracy scores obtained.

Based on these results, it was concluded that performing LDA is not effective for this dataset, but, before deciding to not use feature selection, it was decided to experiment with other state of the art feature analysis techniques, namely Principal Component Analysis (PCA) [TB99]. Some obstacles were identified to realize this experiment. The first approach was to use the PCA class

Table 6.9: Economic Activity Multiclass Classification accuracy scores using LDA (48850 examples)

Classifier	No LDA (baseline)	10 topics	100 topics
Random Forests	0.6787	0.4053	**0.4612**
Bernoulli NB	0.5115	0.4278	0.4278
Multinomial NB	0.4603	-	0.4306
Complement NB	0.5914	0.4157	0.3561
K-Neighbors	0.6283	0.3995	0.4146
SVM (linear)	**0.8075**	**0.4484**	0.4424
Decision Tree	0.6659	0.3320	0.3525
Extra Tree	0.5056	0.3300	0.3419
Random - stratified	0.2504	0.2510	0.2519

which is provided by scikit-learn, but it does not handles sparse matrixes and making the TF-IDF matrix non-sparse is not possible due to memory constraints. The second approach was to use the SparsePCA class which is also provided by scikit-learn, but it also does not handles sparse matrixes. The third and final approach was to use the TruncatedSVD class provided by scikit-learn. This was able to perform the experiment, but it generates a matrix which cannot be computed by some classifiers and, as occurred with LDA, in the other classifiers, the accuracy scores were considerably worse.

As a side note, it was possible to test PCA and SparsePCA using only the first 2000 examples of the dataset with 48850 examples and the accuracy scores were considerably worse. The use of the first 2000 examples is trustable because it generates accuracy scores close (slightly worse) to the ones obtained using the complete dataset.

Recursive Feature Elimination (RFE) trains a classifier multiple times with different features and yields the matrix of features that generated the best classifier according to a given metric. RFE was also tested in conjunction with Complement NB to improve the accuracy score obtained in the baseline, but it was not able to complete in an acceptable amount of time.

Additionally, experiments using Recursive Feature Elimination and Cross-Validated selection (RFECV) approach [GWBV02] were performed. It trains a classifier multiple times with different features and yields the matrix of features that generated the best classifier according to a given metric and using cross-validation. It can also be used to perform predictions using the best trained classifier. RFECV was tested using the first 2000 examples of the dataset with 48950 examples on two classifiers because it takes several time to execute and these first examples can be used as exploratory data because they obtain accuracy scores similar (slightly worse) to the ones obtained using 48850 examples. The first classifier was Complement NB because it is extremely fast to train. It generated an average accuracy score of 68.45% which is better than the 59.14% obtained using the 48850 examples. Because it is not the same number of examples, it is not possible to make a good comparison, but considering that the use of 2000 examples obtains accuracy scores slightly worst than the ones obtained using 48850 examples, the improvement is substantial. In comparison, the macro-F1 score increased from 0.28 to 0.51, a big improvement that confirms that RFECV is beneficial for Complement NB. On the other side, the second classifier was SVM (with linear kernel) and it took a long time to train using the same 2000 examples. For this

classifier, an accuracy score of 78.13% was obtained which is similar to the 80.75% of the baseline, demonstrating that SVM already gives the best coefficients for all features, not requiring further optimization. In comparison, the macro-F1 score decreased from 0.63 to 0.60. This might have happened due to the use of 2000 examples. Otherwise, it indicates that RFECV can be prejudicial if the dataset has an imbalanced distribution.

6.4.2 Impact of synonyms

The following approach was to try the use of a near-synonym dictionary to reduce the amount of features extracted by substituting some lemmas by a synonym. The substitution of the lemmas by synonyms allows to reduce the number of features and obtain less scattered features. Several tools were found for the substitution of near-synonyms, including NLTK and spaCy, but most of them were made for English only and the majority of them were based on Thesaurus.com.

Finally, the diffused wordnet CONTO.PT [Gon16] (version "contopt_0.1_r2_c0.0"), which is an evolution of Onto.PT [GG14], was used to obtain the different synonyms. This wordnet provides a broad set of words and indicates relationships that may exist between them as the possibility of being synonyms, antonyms, hyperonyms, among others. For synonyms, the wordnet presents a set of words and, for each word, a set of synonyms and the trust associated with the synonymy relationship. To perform the substitution of synonyms, a dictionary of words was created in which for each set of synonyms present in CONTO.PT is associated as a synonym for the first word of the set, in order to replace all the words of the set (keys in the dictionary) for the first one (value in the dictionary). For each lemma obtained using StanfordNLP, it was verified whether it was a key in the dictionary and, if it was, it was replaced by the associated value, reducing the number of features obtained.

It is worth noting that this experience accomplishes the substitution of lemmas, so, although there are a large number of substitutions, there are lemmas that are simpler versions than the word to which there is a synonymous, and some substitutions that would be possible are not performed by this reason.

Table 6.10: Economic Activity Multiclass Classification scores using synonyms substitution (79875 examples)

Classifier	Acc@1		Macro-F1@1	
	Baseline	Substituting synonyms	Baseline	Substituting synonyms
Random Forests	0.6196	0.5831	0.36	0.32
SVM (linear)	**0.7435**	**0.7240**	**0.57**	**0.55**

Table 6.10 presents the scores obtained by applying synonyms substitution on the dataset with 79875 examples. As usual, SVM is the one that obtains the best results. By comparing the scores, it is concluded that it lowers the effectiveness of the classifier. The same situation occurs with Random Forests.

Table 6.11 presents the accuracy scores obtained by applying synonyms substitution on the dataset with 48850 examples. Again, SVM is the one with the best scores and it is verified the

Table 6.11: Economic Activity Multiclass Classification accuracy scores using synonyms substitution (48850 examples)

Classifier	Baseline	Substituting synonyms
Random Forests	0.6787	0.6264
Bernoulli NB	0.5115	0.4979
Multinomial NB	0.4603	0.4505
Complement NB	0.5914	0.5508
K-Neighbors	0.6283	0.5967
SVM (linear)	**0.8075**	**0.7863**
Decision Tree	0.6659	0.6221
Extra Tree	0.5056	0.4710
Random - stratified	0.2504	0.2559
SGD	0.7870	0.7847
Bagging	0.7254	0.6896

same situation as in Table 6.10 which shows that substituting synonyms lowers the accuracy of the classifier by about 2%.

These results indicate that some words have a meaning with higher value than others and the use of synonyms prevents that the difference in value can be noticed by the classifiers, lowering them classification capabilities.

Due to the reduction of accuracy and macro-F1 scores, synonyms substitution is not used in further experiments.

6.4.3 Impact of adjectives

An experiment performed to analyze the impact of the removal of adjectives identified by StanfordNLP was performed. This experiment was interesting because strong adjectives are apparently important for the classification task, but other weaker adjectives should not be. Exemplifying, “worst” is a stronger adjective than “bad”. If “worst” occurs in a few complaints restricted to a reduced number of classes and “bad” occurs in many complaints not restricted to a reduced number of classes, examples where “worst” occurred might benefit from the presence of the word to be correctly classified, but examples where “bad” occurred do not benefit from its presence because it makes class separation more difficult. Depending on the amount and type of adjectives present in the dataset, their removal could reduce the amount of features that are irrelevant for the problem.

Table 6.12 presents the scores obtained by removing adjectives on the dataset with 79875 examples. The only case where the removal obtains a better accuracy than the baseline is with SVM to obtain the top-3. In all other cases, the baseline obtains better results, although the difference is not large.

Table 6.13 presents the accuracy scores obtained by removing all adjectives identified by StanfordNLP on the dataset with 48850 examples. Comparing the accuracy scores of all classifiers with the accuracy scores obtained by not removing the adjectives (baseline), the percentage was always the same, differing only on the permillage.

Table 6.12: Economic Activity Multiclass Classification accuracy scores - Comparison of removing adjectives or not (using StanfordNLP) (79875 examples)

Classifier	Baseline	Removing adjectives
Random Forests (Acc@1)	**0.6196**	0.6183
Random Forests (Acc@2)	**0.7840**	0.7811
Random Forests (Acc@3)	**0.8817**	0.8786
SVM (linear) (Acc@1)	**0.7435**	0.7389
SVM (linear) (Acc@2)	**0.8863**	0.8819
SVM (linear) (Acc@3)	0.9451	**0.9452**
Random Forests (Macro-F1@1)	**0.36**	**0.36**
Random Forests (Macro-F1@2)	**0.5**	**0.5**
Random Forests (Macro-F1@3)	**0.59**	0.58
SVM (linear) (Macro-F1@1)	**0.57**	0.56
SVM (linear) (Macro-F1@2)	**0.71**	0.70
SVM (linear) (Macro-F1@3)	**0.77**	**0.77**

These results are indicative that adjectives are partially important for the classifiers, although many of them have a low or even null importance/coefficient. For that reason, the removal of adjectives will not be used on further experiments.

6.5 Over and under sampling

As shown in Table 4.2 and Table 4.1, the distribution of the classes is very unbalanced. To improve the overall classification performance and, more concretely, the performance on the minority classes, we explore two widely used techniques to deal with unbalanced datasets [HG09]: *random under sampling* and *random over sampling*. These balancing techniques remove the capture of data distributions during training, equalizing the number of instances in each class.

Random over sampling is performed by using the RandomOverSampler (ROS) of the Python package "imblearn" [3] and the random under sampling is performed by using the RandomUnderSampler (RUS) of the same package.

There were three alternatives to perform the over sampling: RandomOverSampler, SMOTE and ADASYN. RandomOverSampler duplicates some of the examples of the classes, increasing the number of examples of all classes to the number of examples of the class with the highest number of examples, as indicated in the documentation of "imblearn" [LNOA]. SMOTE generates new samples by interpolation, not distinguishing between easy and hard examples (as indicated in the documentation of "imblearn" [LNOA]). ADASYN generates new samples by interpolation, focusing on generating samples based on the original samples which are wrongly classified using a k-Nearest Neighbors classifier (as indicated in the documentation of "imblearn" [LNOA]). Because we were testing several different classifiers, including a k-Nearest Neighbors classifier, we decided to use the RandomOverSampler to reduce bias in the results.

For the random under sampling, the RandomUnderSampler was chosen to be comparable to the RandomOverSampler. RandomUnderSampler selects randomly a subset of data for the targeted classes (as indicated in the documentation of "imblearn" [LNOA]), reducing the number

[3] https://imbalanced-learn.readthedocs.io/en/stable/

Table 6.13: Economic Activity Multiclass Classification accuracy scores - Comparison of removing adjectives or not (using StanfordNLP) (48850 examples)

Classifier	Baseline	Removing adjectives
Random Forests	0.6787	0.6783
Bernoulli NB	0.5115	0.5119
Multinomial NB	0.4603	0.4612
Complement NB	0.5914	0.5949
K-Neighbors	0.6283	0.6244
SVM (linear)	**0.8075**	**0.8061**
Decision Tree	0.6659	0.6621
Extra Tree	0.5056	0.5054
Random - stratified	0.2504	0.2538

of examples of each class to the number of examples of the class with the smallest number of examples.

Table 6.14: Accuracy and average macro-F1 scores using over and under sampling (79875 examples)

Classifier	Baseline	Random Over Sampling	Random Under Sampling
Random Forests (Acc@1)	0.6196	**0.6548**	0.1515
Random Forests (Acc@2)	0.7840	**0.8072**	0.3216
Random Forests (Acc@3)	0.8817	**0.8953**	0.4268
SVM (linear) (Acc@1)	**0.7435**	0.7362	0.0944
SVM (linear) (Acc@2)	**0.8863**	0.8765	0.1520
SVM (linear) (Acc@3)	**0.9451**	0.9365	0.1952
Random Forests (Macro-F1@1)	0.36	**0.42**	0.06
Random Forests (Macro-F1@2)	0.5	**0.54**	0.12
Random Forests (Macro-F1@3)	0.59	**0.62**	0.17
SVM (linear) (Macro-F1@1)	**0.57**	0.53	0.03
SVM (linear) (Macro-F1@2)	**0.71**	0.64	0.07
SVM (linear) (Macro-F1@3)	**0.77**	0.70	0.13

Table 6.14 presents the accuracy and macro-F1 scores obtained by performing random over sampling and random under sampling on the dataset with 79875 examples. It demonstrates that Random Forests benefits from random over sampling which also indicates that a more balanced dataset would obtain better results. On the other hand, SVM does not benefit of random over sampling nor random under sampling, comproving that it handles very well imbalanced datasets. The random under sampling scores obtained using the dataset with 79875 examples are so low because the technique reduces the amount of examples of each class to only one due to the fact that the class with less training elements is the class VI with only one example. This class should have been removed.

Table 6.15 presents the accuracy and average macro-F1 scores obtained by performing random over sampling and random under sampling on the dataset with 48850 examples. As shown in Table 6.15, when performing random over sampling, the accuracy scores related to Naive Bayes increased considerably and the accuracy of Random Forests also increased, but all other decreased. A similar situation can be observed on the corresponding average macro-F1 score. On the other hand, when performing random under sampling, only the accuracy scores related to the Bernoulli

Table 6.15: Accuracy and average macro-F1 scores using over and under sampling (48850 examples)

Classifier	Accuracy (baseline)	Accuracy ROS	Accuracy RUS	Avg macro--F1 score (baseline)	Avg macro--F1 score ROS	Avg macro--F1 score RUS
Random Forests	0.6787	0.7137	0.4402	0.42	0.49	0.30
Bernoulli NB	0.5115	0.6477	0.4703	0.18	0.48	0.28
Multinomial NB	0.4603	0.7299	0.5223	0.09	0.56	0.37
Complement NB	0.5914	0.7130	0.5258	0.28	0.52	0.37
K-Neighbors	0.6283	0.5456	0.3959	0.46	0.46	0.29
SVM (linear)	**0.8075**	**0.7985**	**0.5555**	**0.63**	**0.62**	**0.43**
Decision Tree	0.6659	0.6294	0.3678	0.45	0.44	0.26
Extra Tree	0.5056	0.4942	0.2074	0.33	0.32	0.15
Random - stratified	0.2504	0.0988	0.1012	0.10	0.07	0.07

NB and the Multinomial NB increased, all the other decreased considerably. Random under sampling on this dataset is performed reducing the amount of examples of each class to 70% of the total of examples of class I because, as indicated in Chapter 4, class VI is removed. Contrary to what happened with the random over sampler, the average macro-F1 score did not follow the pattern of a decrease in accuracy corresponding to a decrease in average macro-F1 score. All average macro-F1 score are relatively low, but 6 of them decreased and 3 of them increased.

6.6 Most representative subclasses

Another experiment was conducted to test if the use of the subclasses III.1 and III.2 as labels could yield better results. Class III is composed by the subclasses III.1, III.2, III.3 and III.4. Subclasses III.1 and III.2 are very large: subclass III.1 has 22967 examples and subclass III.2 has 2371 examples. Subclass III.1 represents 89.95% of the examples of class III and 9.28% of the examples of the dataset. Subclass III.2 represents 28.75% of the examples of class III and 2.97% of the examples of the dataset. Using these two subclasses as labels and maintaining the subclasses III.3 and III.4 within the label "III" reduces the amount of examples classified with the label "III" and might help increasing the accuracy scores and macro-F1 scores by lowering the unbalance of the dataset.

Table 6.16 presents the scores obtained by using the subclasses III.1 and III.2 as labels. For a small difference, almost all the scores of the baseline are better. Only the macro-F1 scores of the SVM are better using the subclasses as labels, which indicates that the use of the subclasses as labels reduces the imbalance between labels.

6.7 Complaint metadata

As will be detailed in Section 6.13, through the analysis of the importances that Random Forests gives to the lemmas obtained from some examples, it was verified that it gave the highest importances to some lemmas that are metadata of the complaints coming from the contact form. Words

Table 6.16: Scores obtained using subclasses III.1 and III.2 as labels

Classifier	Baseline	III.1 and III.2 as classes
Random Forests (Acc@1)	**0.6196**	0.6088
Random Forests (Acc@2)	**0.7840**	0.7754
Random Forests (Acc@3)	**0.8817**	0.8736
SVM (linear) (Acc@1)	**0.7435**	0.7432
SVM (linear) (Acc@2)	**0.8863**	0.8843
SVM (linear) (Acc@3)	**0.9451**	0.9450
Random Forests (Macro-F1@1)	**0.36**	0.35
Random Forests (Macro-F1@2)	**0.5**	0.48
Random Forests (Macro-F1@3)	**0.59**	0.57
SVM (linear) (Macro-F1@1)	0.57	**0.58**
SVM (linear) (Macro-F1@2)	**0.71**	**0.71**
SVM (linear) (Macro-F1@3)	0.77	**0.78**

like “Identificação” (Identification) and “Telefone” (Phone number) had a rather high importance for the classification task, suggesting that the classifier distinguishes whether a complaint came through a contact form or via e-mail to distinguish the class to which the example should belong.

In order to verify if this distinction was important for the classification task, the experiment of removing the metadata of each example was carried out. These metadata were only removed in the complaints coming from the form and were only removed in the specific places where the metadata is found, never throughout the text of the complaint. No token that occurred in the body of the report was removed even if it was a metadata value.

The analysis of the importances was not performed at the level of the SVM, since it provides the importances in the form of coefficients used to distinguish 2 classes, and it is not possible to compare the coefficients since they vary from tuple of classes to tuple of classes.

The following expressions were removed from the examples:

- “Identificação do Queixoso / Denunciante” (Identification of the Complainant)
- “Nome: ” (Name)
- “Morada: ” (Address)
- “Código Postal: ” (Postal Code)
- “Localidade: ” (Locality)
- “Telefone: ” (Phone)
- “E-mail: ”
- “Identificação do Agente Económico visado” (Identification of the target Economic Agent)
- “Identificação dos Factos” (Fact Identification)
- “Data: ” (Date)
- “Local: ” (Location)

- “Descrição: ” (Description)
- “Identificação das Testemunhas” (Identification of Witnesses)

Table 6.17: Scores obtained removing complaint metadata (79875 examples)

Classifier	Baseline	Metadata removed
Random Forests (Acc@1)	0.6196	**0.6346**
Random Forests (Acc@2)	0.7840	**0.7928**
Random Forests (Acc@3)	0.8817	**0.8880**
SVM (linear) (Acc@1)	0.7435	**0.7436**
SVM (linear) (Acc@2)	**0.8863**	0.8855
SVM (linear) (Acc@3)	0.9451	**0.9458**
Random Forests (Macro-F1@1)	0.36	**0.38**
Random Forests (Macro-F1@2)	0.5	**0.51**
Random Forests (Macro-F1@3)	**0.59**	**0.59**
SVM (linear) (Macro-F1@1)	**0.57**	**0.57**
SVM (linear) (Macro-F1@2)	**0.71**	**0.71**
SVM (linear) (Macro-F1@3)	**0.77**	**0.77**

Table 6.17 presents the scores obtained by removing metadata from the examples. Although the difference is very small, the removal of metadata improves the classification task, being recommended. Assuming that the metadata is used to distinguish whether a complaint came through a contact form or via e-mail, the results demonstrate that the presence of metadata is prejudicial to the performance of the classifiers, specially for Random Forests.

The removal of metadata will not be performed in the following experiments so that the comparison with the baseline can be direct.

6.8 Classification conversion

The classification code begins with the 11 fourth-level classes and, recursively, contains the indication of the subclasses to which the complaint belongs, however, there are 4 third-level classes above the fourth-level classes that have been in use in previous experiments. These 4 classes are the “Food” class, “Production”, “Safety” and “Others”.

As pointed in Chapter 4, the “Food” class contains the subclasses I to VII, the “Production” class contains the subclasses VIII to IX, the “Safety” class contains the subclass X and the class “Other”, which in reality was created by us, contains the subclass Z.

As none of the attempts to improve the classification obtained achieved better results than the baseline, it was still tried to train the classifiers to predict the 11 possible classes (as in the baseline) and then convert the prediction into one of the four classes (Food, Production, Safety, Others).

The idea behind this experiment is based on the fact that there are classes with very close concepts in the set of the four classes, being possible that the conversion allows to reduce the number of cases in which the classification is wrong, but, as the expected class and the correct class belong to the same top-level class, the top-level classification would be correct.

If the accuracy score is better than that of the baseline, it makes sense to test a two-tiered approach in which one of the four classes is predicted first, and then it is predicted which of the 11 classes is based on the first prediction.

Table 6.18: Scores obtained converting 4th-level classes into 3rd-level classes (79875 examples)

Classifier	Baseline	Converting class
Random Forests (Acc@1)	0.6196	**0.7225**
SVM (linear) (Acc@1)	0.7435	**0.7841**
Random Forests (Macro-F1@1)	0.36	**0.49**
SVM (linear) (Macro-F1@1)	0.57	**0.62**

Table 6.18 presents the scores obtained by converting the 4th-level classes into 3rd-level classes. The "Converting class" column contains the scores obtained by asking the classifiers trained using the baseline settings to predict the classes of the test subset and then convert the classes before obtaining the scores. It can be verified based on the results in Table 6.18 that a classification to 4 classes gets better accuracy and macro-F1 scores, indicating that a two-tiered approach might obtain good results.

Table 6.19: Confusion matrix of SVM converting 4th-level classes into 3rd-level classes (79875 examples)

		Predicted			
		Food	Production	Safety	Others
Actual	Food	10572	583	1	811
	Production	722	6564	3	914
	Safety	42	173	74	77
	Others	726	1115	6	1574

Table 6.19 presents the confusion matrix obtained converting 4th-level classes into 3rd-level classes. As expected, the main diagonal has the highest values confirming the high accuracy score, but it is interesting that a lot of examples related to class "Others" are being classified as "Food" and "Production", demonstrating that this class is difficult to predict, supposedly due to its diverse nature.

6.9 Pipeline of 3 classifiers

Although the dataset remains quite unbalanced, as the classification conversion experience obtained results superior to the baseline, it was considered that it would be possible to obtain better results by creating a pipeline of 3 classifiers in which each classifier was more specialized.

The first classifier would classify a complaint into one of the four classes and, if the complaint was of the "Food" class, the complaint would be passed to the second classifier which would classify in I to VII, if the complaint was of the class "Production", it would be passed to the third classifier that would classify in VIII to IX and, if the complaint was of classes "Safety" or "Others", it would be already known that it was of class X or Z, respectively, as presented in Chapter 4.

For this experiment, the same training and test subsets were used for all classifiers, but, for the classifiers of the classes I to VII and classes VIII to IX, it was necessary to remove those examples which did not belong to these classes. For that, it was created a list of indexes containing the indexes on the features matrix of the examples that should be removed and the lines with these indexes are removed before passing the training and test subsets to the classifiers.

Table 6.20: Scores obtained using pipeline of 3 classifiers (79875 examples)

Classifier classes	Resampling	Acc@1		Macro-F1@1	
		Random Forests	SVM (linear)	Random Forests	SVM (linear)
Baseline (I to X + Z)	None	0.6196	**0.7435**	0.36	**0.57**
3rd-level classes	None	0.7209	**0.7919**	0.46	**0.62**
Classes I to VII	None	0.8156	**0.9135**	0.42	**0.64**
Classes VIII to IX	None	0.8248	**0.9044**	0.73	**0.88**
Classes I to X + Z	-	0.6037	0.7297	0.35	0.52
3rd-level classes	ROS	0.7250	0.7819	0.50	0.61
Classes I to VII	None	0.8156	**0.9135**	0.42	**0.64**
Classes VIII to IX	None	0.8248	**0.9044**	0.73	**0.88**
Classes I to X + Z	-	0.6108	0.7221	0.36	0.52
3rd-level classes	ROS	0.7250	0.7819	0.50	0.61
Classes I to VII	ROS	0.8223	0.9100	0.43	0.62
Classes VIII to IX	ROS	0.8616	0.9010	0.81	0.87
Classes I to X + Z	-	0.6221	0.7237	0.38	0.55
3rd-level classes	RUS	0.5278	0.6217	0.41	0.5
Classes I to VII	None	0.8156	0.9135	0.42	0.64
Classes VIII to IX	None	0.8248	0.9044	0.73	0.88
Classes I to X + Z	-	0.4738	0.5832	0.25	0.38
3rd-level classes	RUS	0.5278	0.6217	0.41	0.5
Classes I to VII	RUS	0.1858	0.1413	0.09	0.05
Classes VIII to IX	RUS	0.8437	0.8917	0.82	0.87
Classes I to X + Z	-	0.2421	0.2909	0.15	0.18

Table 6.20 presents the scores obtained by applying or not random over sampling (ROS) or random under sampling (RUS) when using some classifiers and using a pipeline of 3 classifiers. Using the pipeline of classifiers without resampling obtained the best results with SVM, but the accuracy and macro-F1 scores are not better than the baseline. Performing random over sampling allowed to obtain slightly better accuracy scores, except on the classifier of the 3rd-level classes, but that was not enough to improve the classification of the classes I to X + Z. Performing random under sampling reduced drastically the scores, but that is related to the number of samples for each class being lowered to one.

Most probably, this approach presents limitations because the errors performed by the classifier of the 3rd-level classes are propagated to the final statistics generated for the classes I to X + Z. To check if the errors made by the classifier of the 3rd-level classes are a limiting factor, it would be necessary to assume that the 3rd-level class was known (situation where there are no errors from the corresponding classifier) and use the classifiers of the 4th-level classes to make predictions, test that was not performed. If looking at the top-2 of the classifier of the 3rd-level classes and then obtaining the top-1 of the classifiers of the 4th-level classes, there would be 2 possible classes to choose and the scores should improve considerably in relation to the top-2 of the baseline.

6.10 SVM optimization

Experiments performed on the dataset with 48850 examples to increase the accuracy of SVM (with linear kernel) generating different class weights and balanced class weights [HG09] obtained accuracy and average macro-F1 scores approximated to the ones obtained using the default parameters. Using different class weights, by performing several iterations of randomly choosing values between 0 and 10 for all class weights, was possible to obtain an accuracy score of 80.96%, slightly higher than the 80.75% of the baseline and was possible to obtain an average macro-F1 of 0.63 which is equal to the one of the baseline. Using balanced class weights was possible to obtain an accuracy score of 80.73%, slightly lower than the 80.75% of the baseline and was possible to obtain an average macro-F1 of 0.64 which is slightly higher than the 0.63 of the baseline.

Table 6.21: Scores obtained using different kernels for SVM (48850 examples)

Kernel	Acc@1	Macro-F1@1	Time (s)
linear	**0.7988**	**0.59**	7663
poly	0.6406	0.38	47880
rbf	0.7843	0.54	18090
sigmoid	0.7927	0.56	6318

SVM has different kernels available: linear, poly, rbf, sigmoid, precomputed. All of them were tested to discover which one is best. Table 6.21 presents the scores obtained using the different kernels on the dataset with 48850 examples. Only the "precomputed" kernel is not shown because it gave an error indicating that it does not support sparse features matrices. Both in terms of accuracy and macro-F1 scores, the best kernel is the linear kernel, followed by the sigmoid kernel in second place. The linear kernel is fast and presents the best accuracy and average macro-F1 scores, so it was our choice.

6.11 Additional experiments

6.11.1 StanfordNLP PoS tagging

StanfordNLP was configured using the pipeline 'tokenize,mwt,pos,lemma,deparse' which tokenizes the text, expands multi-word tokens into multiple words, performs part-of-speeach tagging, obtains lemmas and, finally, performs dependency parsing. Dependency parsing was performed, but the data extracted was not used, so it can be safely ignored. Because the use of PoS was mainly important to detect adjectives and punctuation and adjectives will not be removed and punctuation can be removed by comparing the tokens to the Python list of punctuation, it was necessary to verify if PoS is necessary to obtain lemmas. If not necessary, the preprocessing time would be lower. By parsing the text of one large example, using the pipeline 'tokenize,mwt,lemma', 218 lemmas were identified and, using the pipeline 'tokenize,mwt,pos,lemma', 217 lemmas were identified, so PoS is relevant.

6.11.2 Stanford CoreNLP

Stanford CoreNLP is the Java version of StanfordNLP. To verify if it was possbile to improve the preprocessing time, Stanford CoreNLP was tested using it as a server and the client implemented in Python as the rest of the prototype. The test was performed using the English models for CoreNLP on the 2000 first examples of the dataset with 48850 examples. On both, dependency parsing was not enabled, they only did all the steps necessary between tokenization and lemmatization. The execution time moved from 1.08 doc/s to 7.39 doc/s which is an increase of about 7 times. Additionally, StanfordNLP used 4 threads without any specific configuration for that, but CoreNLP used only 1 thread although 16 were allowed. This test was not made using a model for Portuguese for CoreNLP because no official version was found and, due to that reason, the Python version was maintained in use.

6.12 Error analysis

Based on the different accuracy and average macro-F1 scores obtained, it was decided to focus on SVM for the sake of error analysis. To better understand in which situations the classifier was making erroneous predictions, it was decided to randomly sample 50 different examples from the dataset where the classifier was not capable of correctly predict (from the top-3 predictions) the gold-standard class.

To perform this experiment, a REST service was implemented which receives a text and the trained classifier name to use. The service performs the same preprocessing, feature extraction and feature selection tasks that were used to train the classifier and returns the importances/coefficients given by the classifier to the features present in the text and the set of classes sorted from highest probability to lowest.

Through the manual analysis of such cases, the following observations were performed:

- Short text, not providing enough information to classify the economic activity.
- Classes that overlap semantically (to a certain degree), confusing the classifier. For example, class VIII that apparently overlaps with the classes II and V.
- Text that is about a specific class but contains words that are highly related to different classes. An example is the case where the text mentions that an establishment put fruit (indicating bananas, apples...) on the street floor. Looking at the names of the fruits, the classifier may point to the food area instead of pointing to the area of safety and environment.
- The text is not in the same language as the one configured. Two cases were found where the text was in English instead of Portuguese. The classifier failed completely the prediction.
- Text mixing information that is important to understand the complaint.

- Text that does not contain a complaint, but provides or requests new information. An example is the case where someone that has made a complaint before and simply wants to know the state of it.
- Small amount of text that does not contain a complaint, but contains a few meta-data or indicates that the complaint is attached.
- Text that is a complement to a previous complaint, but that does not contain the previous complaint, not being possible to obtain any information related to the economic activity from it.
- Possibility of human error on the classification. An example exposed a situation of lack of structural security and hygiene and the class IX was assigned, but the class X is, apparently, the most correct.
- Text related to a complaint followed by some content in English. An example of that are complaints received by e-mail that present a few text indicating to think twice before printing the e-mail content.

6.13 Observations

Beyond the analysis of errors performed, it was performed an analysis of which information was used by Random Forests to generate predictions. Random Forests was used, instead of SVM, only because it shows only one set of importances and SVM shows coefficients for every pair of classes. The following observations were realized during the analysis of examples:

- Some punctuation draws attention to the indignation and severity of the complaint, but it is not used by the classifier because the punctuation is removed in the preprocessing.
- The classifier gives a very large (from the highest) importance to some words that are meta-data of the complaints (“e-mail”, “locality”, “telephone”...). This does not make much sense as it occurs in most complaints, so it should not be used to discriminate against classes. The assumption is that the classifier distinguishes between form and e-mail to generate the prediction.
- There are several features present with zero importance, that is, they are ignored by the classifier.
- The classifier does not use specific data (such as company name or email) as features, which makes sense, as it may have never seen this information.
- Several words are used as important (ordered from most to least important: “encomenda” (order), “enviar” (send), “visar” (aim), “pagar” (pay), “queixoso” (complainant), “site”, “receber” (receive), “denunciar” (report), “entregar” (deliver), “devolver” (return), “queixa”

(complaint)...) and make sense to identify the class of the complaint (class VII - Distance selling, in this case).

- No syntactic dependency was found that could be used to improve the functioning of the classifier.

6.14 Conclusions

From all experiments, the most promising configuration using the dataset with 79875 examples is the use of StanfordNLP to obtain all the lemmas, then remove punctuation and stop words, obtain features using TF-IDF and, to complete, train a classifier using SVM with the linear kernel. This configuration obtains a maximum accuracy score of 74.35%. This value could be improved by using NLTK, removing metadata and/or, depending on the task, using examples that are only of ASAE responsibility. The last option is not usable in the scope of this thesis because the responsibility is not known, but it can be used for other scopes. The removal of metadata is encouraged because it allowed a small increase of 0.01% in accuracy and, in conjunction with other improvements, the difference might be noticeable.

The 0.9451 score obtained with SVM and top-3 on the dataset with 79875 examples demonstrates that presenting a set of classes sorted by confidence will be an effective help.

Through the analysis of the different experiments, it is possible to conclude that SVM is the best classifier to use without resorting to deep learning architectures. It is expected that the accuracy score will be higher when training a new classifier using the same dataset with new examples added because the amount of examples will be higher and the risk of human error will be reduced through the comparison of the prediction of the classifier with the decision of the human operator. Also, this research provided several insights that can be used later in the project to adapt the content of the complaints in order to reach more accurate predictions.

The accuracy scores of both the dataset with 48850 examples and the dataset with 79875 examples did not differ much because these datasets have approximately the same distribution of classes.

Finally, the use of ensembles (Bagging classifier) based on decision trees, which usually have interesting performances, provided accuracy scores higher than the ones obtained using Random Forests, but considerably lower than the ones provided by SVM.

For the imbalanced complaints dataset that we have used, SVM with a linear kernel proved to be the best option among the experimented models. It is also reasonably fast, allows to get probability scores and gives the best accuracy scores and average macro-F1 scores. It is specially valuable if a ranking output makes sense, given the high accuracy obtained when considering the top-3 predicted classes.

Chapter 7

Infractions Prediction

The second task that was performed is related to obtaining the infractions that may be associated with a given complaint. This task has the difficulty that there may be several or no infractions associated with each complaint, being a multilabel classification task. In view of the large number of infractions possible, it was decided that a multilabel classification would not compensate due to the difficulty in distinguishing the classes. In that regard, it was decided to classify each complaint on the basis of the most serious infraction. Instantiating, as pointed in Chapter 4, there are 3 possible classes: “Crime”, “Misdemeanor” and “Other”. The class “Crime” indicates that the complaint indicates at least one crime, and there may be misdemeanors. The class "Misdemeanor" indicates that the complaint indicates at least one misdemeanor, but no crime. The class “Other” indicates that the complaint does not indicate any crime or misdemeanor. This classification in “Crime”, “Misdemeanor” and “Other” is a single-label multiclass classification problem, partially following the same approach that was followed in Chapter 6.

This chapter will not present a detailed rationale of each experience, because it is the same as for the corresponding experience in Chapter 6. The notation present in the tables is also equivalent to that used in Chapter 6.

7.1 Baseline

To obtain the baseline, StanfordNLP was used in the preprocessing and 1-grams TF-IDF was used to obtain the data representation. StanfordNLP is used to perform the preprocessing on all the following experiments.

Table 7.1: Infraction Multiclass Classification scores for top-k predictions

Classifier	Acc@1	Acc@2	Macro-F1@1	Macro-F1@2
Random Forests	0.6902	0.9576	0.56	0.78
SVM (linear)	**0.7341**	**0.9738**	**0.68**	**0.9**

Table 7.2: Confusion matrix of the baseline SVM (Top-1)

		Predicted		
		Misdemeanor	Crime	Other
Actual	Misdemeanor	9200	99	2727
	Crime	442	579	258
	Other	2695	149	7808

It can be seen in Table 7.1 that Random Forests gets acceptable results by correctly predicting 69.02% of the examples of the test subset. Looking at the top-2, Random Forests reaches 95.76% of accuracy, but this value is high because only 5.4% of the examples are labeled as "Crime". For SVM (linear), as observed in Chapter 6, it obtains the best accuracy and macro-F1 scores, predicting correctly 73.41% of the examples and reaching 97.38% for the top-2. Analyzing the confusion matrix in Table 7.2 that was obtained for SVM accepting the top-1, it is concluded that most of the examples of the different classes are correctly classified, supporting the accuracy score obtained. This is especially important for examples of class "Crime", as they are the most important examples for ASAE and whose identification is a priority.

7.2 Feature extraction

7.2.1 Data representation techniques

Table 7.3: Infraction Multiclass Classification scores using different data representation techniques

Classifier	Count	TF-IDF (baseline)	BERT
Random Forests (Acc@1)	0.**7066**	0.6902	0.6328
Random Forests (Acc@2)	**0.9598**	0.9576	0.9492
SVM (linear) (Acc@1)	0.6754	**0.7341**	-
SVM (linear) (Acc@2)	-	**0.9738**	-
Random Forests (Macro-F1@1)	**0.58**	0.56	0.46
Random Forests (Macro-F1@2)	**0.8**	0.78	0.69
SVM (linear) (Macro-F1@1)	0.62	**0.68**	-
SVM (linear) (Macro-F1@2)	-	0.9	-

Table 7.3 indicates that the count technique is slightly better for Random Forests, but that TF-IDF is a considerably better option to use with SVM (linear). BERT obtains the worst results for all tested cases. The same conclusions were obtained in Chapter 6. The count technique was tested with probabilities disabled due to the amount of time necessary to execute the experiments, so it is not possible to present the top-2. The hashing technique was not tested because, based on the findings in Chapter 6, it obtains worst results than the count technique and it was used on that chapter to be compared with the count technique. Several tries were performed to use BERT, but due to the amount of time it takes to run and due to problems of resource starvation when the execution was almost finishing, the results for SVM could not be obtained.

On the following experiments, TF-IDF will be used to obtain the data representation because it is the one that obtains the best results (using SVM).

7.2.2 Impact of n-grams

Table 7.4: Infraction Multiclass Classification scores using different n-grams

Classifier	1-gram	1 to 2-grams	2-grams	1 to 3-grams	2 to 3-grams	3-grams
Random Forests (Acc@1)	**0.6902**	0.6834	0.6621	0.6684	0.6482	0.6293
Random Forests (Acc@2)	0.9576	0.9568	**0.9583**	0.9570	0.9579	0.9575
SVM (linear) (Acc@1)	0.7341	**0.7474**	0.7294	0.7463	0.7245	0.6899
SVM (linear) (Acc@2)	**0.9738**	0.9737	0.9682	0.9727	0.9669	0.9607
Random Forests (Macro-F1@1)	**0.56**	0.54	0.53	0.53	0.52	0.5
Random Forests (Macro-F1@2)	**0.78**	0.77	**0.78**	0.77	**0.78**	**0.78**
SVM (linear) (Macro-F1@1)	0.68	**0.69**	0.67	**0.69**	0.65	0.6
SVM (linear) (Macro-F1@2)	**0.9**	**0.9**	0.87	**0.9**	0.86	0.81

Table 7.4 presents the scores obtained using different intervals of n-grams. The results are not completely conclusive because the best results do not appear always on the same classifier nor using the same top-k, being highly dependent of the context of use.

The following experiments will use 1-grams TF-IDF because it presents the best overall accuracy and macro-F1 scores (using SVM).

7.3 Feature selection

7.3.1 Latent Dirichlet Allocation (LDA)

Table 7.5: Infraction Multiclass Classification scores using LDA

Classifier	No LDA (baseline)	10 topics	100 topics
Random Forests (Acc@1)	**0.6902**	0.4998	0.5360
Random Forests (Acc@2)	**0.9576**	0.9218	0.9458
SVM (linear) (Acc@1)	**0.7341**	0.5103	0.5122
SVM (linear) (Acc@2)	**0.9738**	0.9466	0.9468
Random Forests (Macro-F1@1)	**0.56**	0.37	0.4
Random Forests (Macro-F1@2)	**0.78**	0.69	0.7
SVM (linear) (Macro-F1@1)	**0.68**	0.29	0.31
SVM (linear) (Macro-F1@2)	**0.9**	0.65	0.65

Table 7.5 presents the scores obtained performing LDA on the features matrix obtained using 1-grams TF-IDF and then passing the result to a classifier. LDA is a clustering technique which tries to group the features with the same properties and divide them from the others. The table indicates that without a doubt the use of LDA is prejudicial to the classification task creating the number of topics/groups defined. Increasing the amount of topics allowed a better performance of LDA, but trying larger numbers of topics would be, in the limit, like not using LDA because the number of features per component would decrease until each component has one feature.

Due to the reason explained, LDA will not be used on the following experiments.

Table 7.6: Infraction Multiclass Classification scores using synonyms substitution

Classifier	Baseline	Substituting synonyms
Random Forests (Acc@1)	**0.6902**	0.6709
Random Forests (Acc@2)	**0.9576**	0.9544
SVM (linear) (Acc@1)	**0.7341**	0.7213
SVM (linear) (Acc@2)	**0.9738**	0.9683
Random Forests (Macro-F1@1)	**0.56**	0.53
Random Forests (Macro-F1@2)	**0.78**	0.75
SVM (linear) (Macro-F1@1)	**0.68**	0.65
SVM (linear) (Macro-F1@2)	**0.9**	0.88

7.3.2 Impact of synonyms

Table 7.6 presents the scores obtained by performing substitution of synonyms. In all cases, the baseline is better than the use of substitution. As explained in Chapter 6, this indicates that some words have a meaning with higher value than others and the use of synonyms prevents that the difference in value can be noticed by the classifiers, lowering them classification capabilities.

Because the substitution of synonyms does not obtain better scores, it will not be used on the following experiments.

7.3.3 Impact of adjectives

Table 7.7: Infraction Multiclass Classification scores - Comparison of removing adjectives or not (using StanfordNLP)

Classifier	Baseline	Removing adjectives
Random Forests (Acc@1)	0.6902	**0.6921**
Random Forests (Acc@2)	**0.9576**	0.9564
SVM (linear) (Acc@1)	**0.7341**	0.7316
SVM (linear) (Acc@2)	**0.9738**	0.9722
Random Forests (Macro-F1@1)	**0.56**	0.55
Random Forests (Macro-F1@2)	**0.78**	0.77
SVM (linear) (Macro-F1@1)	**0.68**	0.67
SVM (linear) (Macro-F1@2)	**0.9**	**0.9**

Table 7.7 presents the scores obtained by performing removal of adjectives. Except for the top-1 with Random Forests, with a marginal difference, the baseline is better than the use of removal. As explained in Chapter 6, this indicates that adjectives are important to differentiate the different classes and the use of adjectives with higher value than others might indicate one type of infraction instead of other, happening that the removal of adjectives prevents that the difference in value can be noticed by the classifiers, lowering the classification capabilities.

Because the removal of adjectives does not obtain better scores, it will not be used on the following experiments.

Table 7.8: Infraction Multiclass Classification scores using over and under sampling

Classifier	Baseline	Random Over Sampling	Random Under Sampling
Random Forests (Acc@1)	**0.6902**	0.6872	0.6151
Random Forests (Acc@2)	0.9576	**0.9613**	0.8671
SVM (linear) (Acc@1)	**0.7341**	0.7301	0.6585
SVM (linear) (Acc@2)	**0.9738**	0.9694	0.9328
Random Forests (Macro-F1@1)	0.56	**0.58**	0.55
Random Forests (Macro-F1@2)	0.78	**0.81**	**0.81**
SVM (linear) (Macro-F1@1)	**0.68**	0.67	0.6
SVM (linear) (Macro-F1@2)	**0.9**	0.87	0.87

7.4 Over and under sampling

Table 7.8 presents the scores obtained by performing random over sampling and random under sampling. For Random Forests, the use of random over sampling depends if what is intended is the top-1 or top-2, but its use is acceptable due to the marginal difference for the top-1 between using or not. For SVM, no resampling should be made because it already handles well the imbalance of the dataset.

7.5 Conclusions

The configuration that maximizes the accuracy score is the use of StanfordNLP to obtain all the lemmas, then remove punctuation and stop words, obtain features using TF-IDF and, to complete, train a classifier using SVM with the linear kernel. This configuration obtains a maximum accuracy score of 73.41% and improves to 97.38% looking at the top-2, as explained in Section 7.1.

Analyzing Table 4.3, it is noticeable that it is important to take into account the proportion of examples of each class because it confirms that between accepting the top-1 and top-2 there is a large difference in the accuracy score obtained.

Chapter 8

Conclusions and Future Work

Throughout the work, different experiences were carried out that allowed answering the different research questions.

To find out the best NLP techniques to preprocess complaints, it was tested the use of NLTK, StanfordNLP and spaCy, concluding that NLTK is the best option for most classifiers, but that the difference between the scores of the tools is small, being used StanfordNLP which provides a more complete support for more in-depth experiments.

In order to find out the best techniques of data representation, it was tested the use of Count, Hashing, TF-IDF and BERT and it was concluded that TF-IDF is the best option in conjunction with SVM, followed by Count with Random Forests.

Regarding the text categorization algorithms, several were tested, concluding that SVM was always the one that obtained better results if it was able to converge. The features should be extracted using the data representation technique and the amount of n-grams that fits best the text categorization algorithm in use and no feature selection should be performed.

Finally, the use of text categorization algorithms as a recommendation to humans by providing labels in the form of a ranking obtained very good results because, even with 11 possible labels, the top-3 of the baseline reaches an accuracy score of 94.51%.

With this work, it was possible to obtain a configuration that allows very promising results and that can be integrated into the IA.SAE project and can be refined in order to obtain even better results.

From all the experiments performed it was possible to obtain a few insights that indicate that the configurations with less steps are the ones that obtain the best results. Detailing, it was concluded that, for preprocessing, simply perform tokenization, ignoring punctuation, and get lemmas and remove those that are stop words will provide good results. In terms of feature extraction, TF-IDF is the best tested way of representing the data. And, finally, for datasets like ours, with an imbalanced distribution of classes, SVM with a linear kernel proved to be the best option among the experimented models. It is reasonably fast, allows to get probabilities, from which it is possible to create a ranking, and gives the best accuracy and average macro-F1 scores. It is specially

valuable if we want a ranking because the accuracy score for the prediction of the economic activity considering the top-3 reaches the 94.51% which is a very high accuracy comparing it with the top-1 which reaches the 74.35%. It is interesting to discover that using additional techniques like feature selection and different quantities of n-grams does not obtain better accuracy scores.

For the prediction of the most severe infraction, the best configuration was the same as for the prediction of the economic activity, obtaining an accuracy score of 73.41% for the top-1 and increasing to 97.38% for the top-2.

This work provided insights on which techniques might be pursued in similar tasks and suggests new experiments that might improve the classification tasks.

Some improvements that could be realized in future iterations were identified throughout the work. The first possible improvement corresponds to making an effort to ignore or translate texts in languages other than Portuguese because some texts have been found in English and there are not enough examples to support a multilingual classifier. The second possible improvement corresponds to discarding texts without enough information to be considered valid complaints. An example of this type of text is an empty text that is automatically ignored by the current system because it can not be preprocessed. The third possible improvement corresponds to try to correct the words present in the texts because, throughout the experiments, it has been noticed that there are mistakenly written words that are used as features. These words are not problematic because the classifiers tend to give them an insignificant importance, but it would be possible to use approximation measures or automatic spell checkers to search for and correct these words, reducing the number of features and increasing the number of occurrences of the features corresponding to the correct version of the word.

In addition to these improvements, classifiers can be created to predict the subclasses of each class. This can be done based on the work done in this thesis and other techniques of word embeddings and deep learning can be used. Finally, the results obtained can be compared with the results presented in this thesis.

This thesis contributed to obtain an easily adaptable framework capable of using different algorithms to classify texts. This framework will be used to identify the economic activity of the target entity of the complaint and discover if a complaint presents an infringement and its type. The framework is relatively generic and can be used as a basis for other experiments, inclusively experiments related to other use cases or modules of the IA.SAE project, but it will be fine-tuned for final integration into the project.

References

[ABV18] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[AJ18] Deepti Ameta and Pokhar Mal Jat. Information extraction from wikipedia articles using DeepDive. In *2018 International Conference on Communication information and Computing Technology (ICCICT)*, pages 1–6. IEEE, February 2018.

[APA+17] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. Technical report, University of Georgia, 2017.

[BCJ+18] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BLB+13] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[BT15] Bogdan Batrinca and Philip C. Treleaven. Social media analytics: a survey of techniques, tools and platforms. *AI & Society*, 30(1):89–116, February 2015.

[CGR18] Surbhit Chugani, K. Govinda, and Somula Ramasubbareddy. Data Analysis of Consumer Complaints in Banking Industry using Hybrid Clustering. In *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, pages 74–78. IEEE, February 2018.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

REFERENCES

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DN18] Gerardo Ocampo Diaz and Vincent Ng. Modeling and prediction of online product review helpfulness: A survey. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 698–708, 2018.

[FB16] Ana Catarina Forte and Pavel B. Brazdil. Determining the level of clients' dissatisfaction from their commentaries. In *Computational Processing of the Portuguese Language - 12th International Conference, PROPOR 2016*, volume 9727 of *Lecture Notes in Computer Science*, pages 74–85. Springer, 2016.

[FD95] Ronen Feldman and Ido Dagan. Knowledge Discovery in Textual Databases (KDT). Technical report, Bar-Ilan University, 1995.

[FK14] Ahmad Fauzan and Masayu Leylia Khodra. Automatic Multilabel Categorization using Learning to Rank Framework for Complaint Text on Bandung Government. In *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, pages 28–33. Institut Teknologi Bandung, IEEE, 2014.

[GG14] Hugo Gonçalo Oliveira and Paulo Gomes. ECO and Onto.PT: A flexible approach for creating a Portuguese wordnet automatically. *Language Resources and Evaluation Journal*, 48(2):373–393, 2014.

[Gol15] Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing. Technical report, Bar-Ilan University, 2015.

[Gon16] Hugo Gonçalo Oliveira. CONTO.PT: Groundwork for the Automatic Creation of a Fuzzy Portuguese Wordnet. In *Proceedings of 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *LNAI*, pages 283–295, Tomar, Portugal, July 2016. Springer.

[GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, January 2002.

[HG09] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[KZYY18] Feyzullah Kalyoncu, Engin Zeydan, Ibrahim Onuralp Yigit, and Ahmet Yildirim. A Customer Complaint Analysis Tool for Mobile Network Operators. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 609–612. IEEE, August 2018.

[Li14] Hang Li. *Learning to rank for information retrieval and natural language processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publ., San Rafael, CA, 2 edition, 2014.

[LNOA] G. Lemaitre, F. Nogueira, D. Oliveira, and C. Aridas. Over-sampling - imbalanced-learn 0.4.3 documentation. `https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html#ill-posed-examples`. Last accessed on April 02, 2019.

[MCD15] Elaheh Momeni, Claire Cardie, and Nicholas Diakopoulos. A survey on assessment and ranking methodologies for user-generated content on the web. *ACM Comput. Surv.*, 48(3):41:1–41:49, December 2015.

[Mit97] Tom M. (Tom Michael) Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[MRS09] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University, 2009.

[MS99] Christopher D. Manning and Hinrich. Schutze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[OT17] Fouad Nasser A Al Omran and Christoph Treude. Choosing an nlp library for analyzing software documentation: A systematic literature review and a series of experiments. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, pages 187–197, Piscataway, NJ, USA, 2017. IEEE Press.

[OTB+14] Francisco Villarroel Ordenes, Babis Theodoulidis, Jamie Burton, Thorsten Gruber, and Mohamed Zaki. Analyzing Customer Experience Feedback Using Text Mining: A Linguistics-Based Approach. *Journal of Service Research*, 17(3):278–295, 2014.

[PKF+13] Gerald Petz, Michał Karpowicz, Harald Fürschuß, Andreas Auinger, Václav Stříteský, and Andreas Holzinger. Opinion mining on the web 2.0 – characteristics of user generated content and their impacts. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, pages 35–46. Springer, 2013.

[QDZM18] Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[RND09] Stuart J. (Stuart Jonathan) Russell, Peter Norvig, and Ernest Davis. *Artificial intelligence: A Modern Approach*. Pearson, 3 edition, 2009.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[SS14] Carson Sievert and Kenneth Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[SZ15] Shuang Dong and Zhihong Wang. Evaluating service quality in insurance customer complaint handling throught text categorization. In *2015 International Conference on Logistics, Informatics and Service Sciences (LISS)*, pages 1–5. IEEE, July 2015.

[TB99] Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

REFERENCES

[ZJN18] Zuraini Zainol, Mohd T. H. Jaymes, and Puteri N. E. Nohuddin. VisualUrText: A Text Analytics Tool for Unstructured Textual Data. *Journal of Physics*, 2018.

Appendix A

Stop words

List of stop words provided by NLTK and used in all experiments: “serei”, “numa”, “houveremos”, “não”, “dele”, “uma”, “tuas”, “quando”, “entre”, “teríamos”, “mesmo”, “se”, “esses”, “a”, “tenha”, “teremos”, “dos”, “mas”, “houverão”, “pela”, “vocês”, “estivéramos”, “e”, “seja”, “tiver”, “aqueles”, “tivermos”, “aquela”, “houverei”, “fôssemos”, “éramos”, “o”, “terei”, “estiveram”, “era”, “fora”, “teriam”, “tiveram”, “estejamos”, “houvesse”, “do”, “houverá”, “seriam”, “estivemos”, “estavam”, “seríamos”, “teve”, “sua”, “você”, “tinham”, “fui”, “os”, “lhe”, “qual”, “estávamos”, “fomos”, “vos”, “sem”, “quem”, “isto”, “houveram”, “tenho”, “fôramos”, “que”, “houvermos”, “pelas”, “meu”, “muito”, “estes”, “seremos”, “houvemos”, “estou”, “na”, “estive”, “tinha”, “houver”, “estivera”, “fossem”, “de”, “tém”, “à”, “com”, “também”, “tua”, “nossas”, “temos”, “delas”, “teria”, “tu”, “tivemos”, “esteja”, “houvera”, “esse”, “depois”, “essa”, “foi”, “esta”, “tivera”, “aos”, “nossa”, “tenhamos”, “for”, “houve”, “até”, “esteve”, “houveriam”, “foram”, “estejam”, “tivessem”, “houveria”, “estivermos”, “nem”, “eu”, “em”, “aquilo”, “seus”, “terá”, “só”, “teu”, “já”, “nos”, “nas”, “eles”, “minha”, “te”, “nosso”, “formos”, “às”, “tive”, “hajam”, “está”, “tem”, “lhes”, “estas”, “para”, “suas”, “são”, “forem”, “como”, “teus”, “seu”, “ela”, “num”, “houveríamos”, “somos”, “será”, “havemos”, “há”, “meus”, “ou”, “houvéramos”, “nós”, “aquelas”, “houvessem”, “houverem”, “tínhamos”, “deles”, “este”, “ele”, “pelos”, “estiverem”, “serão”, “dela”, “estivessem”, “eram”, “sou”, “no”, “nossos”, “essas”, “tivéramos”, “um”, “tiverem”, “me”, “sejamos”, “elas”, “isso”, “das”, “terão”, “houvéssemos”, “por”, “estivesse”, “hajamos”, “mais”, “aquele”, “haja”, “estivéssemos”, “hei”, “hão”, “minhas”, “tivesse”, “estão”, “estava”, “tivéssemos”, “da”, “ao”, “as”, “sejam”, “estiver”, “tenham”, “pelo”, “fosse”, “estamos”, “seria”.

Stop words

Appendix B

Paper submitted to the 7th International Conference on Statistical Language and Speech Processing

Automatic Identification of Economic Activities in Complaints

Luís Barbosa[1], João Filgueiras[1,2], Gil Rocha[1,2][0000−0001−8252−7292], Henrique Lopes Cardoso[1,2][0000−0003−1252−7515], Luís Paulo Reis[1,2][0000−0002−4709−1718], João Pedro Machado[3], Ana Cristina Caldeira[3], and Ana Maria Oliveira[3]

[1] Departamento de Engenharia Informática,
Faculdade de Engenharia da Universidade do Porto

[2] Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC)
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

[3] Autoridade de Segurança Alimentar e Económica (ASAE)
Rua Rodrigo da Fonseca, 73, 1269-274 Lisboa, Portugal

{up201405729, gil.rocha, filgueiras, hlc, lpreis}@fe.up.pt
{jpmachado, accaldeira, amoliveira}@asae.pt

Abstract. In recent years, public institutions have undergone a progressive modernization process, bringing several administrative services to be provided electronically. Some institutions are responsible for analyzing citizen complaints, which come in huge numbers and are mainly provided in free-form text, demanding for some automatic way to process them, at least to some extent. In this work, we focus on the task of automatically identifying economic activities in complaints submitted to the Portuguese Economic and Food Safety Authority (ASAE), employing natural language processing (NLP) and machine learning (ML) techniques for Portuguese, which is a language with few resources. We formulate the task as several multi-class classification problems, taking into account the economic activity taxonomy used by ASAE. We employ features at the lexical, syntactic and semantic level using different ML algorithms. We report the results obtained to address this task and present a detailed analysis of the features that impact the performance of the system. Our best setting obtains an accuracy of 0.8164 using SVM.When looking at the three most probable classes according to the classifier's prediction, we report an accuracy of 0.9474.

Keywords: Text Categorization · Natural language processing · User-generated text · Complaint analysis

1 Introduction

Several countries have public administration institutions that provide public services electronically. Moreover, such institutions are responsible for processing citizen requests, also performed by electronic means, often materialized through

email contacts or by filling-in contact forms in so-called virtual counters. In specific types of public institutions, such as those in charge of enforcing compliance of citizens or economic agents, a significant number of such requests are in fact complaints that need to be appropriately dealt with.

The amount of complaints received can reach the thousands in a short period of time, depending on the size of the country/administrative region. The Portuguese Economic and Food Safety Authority (ASAE), for instance, receives more than 20 thousand complaints annually, more than 30% of which are usually found not to be in the jurisdiction of ASAE; the rest are sent to the ASAE Operational Units. Given the high amount of complaints, the use of human labor to analyze and properly handle them quickly becomes a bottleneck, bringing the need to automate this process to the extent possible. One of the obstacles to do it effectively is the fact that contact forms typically include free-form text fields, bringing high variability to the quality of the content written by citizens.

This work focuses on automatically identifying economic activities in complaints written in Portuguese, through the use of natural language processing (NLP) and machine learning (ML) techniques. Portuguese is a low-resourced language in terms of NLP. We employ different features and analyze which ones give the best results using different ML algorithms. We start by discussing related work in Section 2. Section 3 describes the dataset used in this work. We detail the employed preprocessing and feature extraction techniques in Section 4. Using different ML models, Section 5 describes several experiments, including those related with feature selection and data balancing techniques. In Section 6, we provide an error analysis and make pertinent observations on the difficulty of the task. Finally, Section 7 concludes and presents some lines of future work.

2 Related Work

Although several works exist on analyzing user-generated content, they mostly study social media data [1], focusing on tasks such as sentiment analysis and opinion mining [15], or predicting the usefulness of product reviews [4]. Forte and Brazdil [6] focus on sentiment polarity of Portuguese comments, and use a lexicon-based approach enriched with domain specific terms, formulating specific rules for negation and amplifiers. Literature on (non-social media) complaint analysis is considerably more scarce, mainly due to the fact that such data is typically not publicly available. Nevertheless, the problem has received significant attention from the NLP community, as a recent task on consumer feedback analysis shows [11]. Given the different kinds of analysis one may want to undertake, however, the task concentrates on a single goal: to distinguish between comment, request, bug, complaint, and meaningless. In our work, we want to further analyze the contents of complaints, with a finer granularity.

Ordenes et al. [14] propose a framework for analyzing customer experience feedback, going beyond sentiment analysis and using a linguistics-based text mining model. The approach explores the identification of activities, resources and context, so as to automatically distinguish compliments from complaints, re-

garding different aspects of the customer feedback. This is made possible through a manual annotation process. The work focuses on a single activity domain, and in the end aims at obtaining a refined sentiment analysis model. In our case, we aim at distinguishing amongst a number of economic activities, without entering into a labor-intensive annotation process of domain-specific data.

Traditional approaches to text categorization employ feature-based sparse models, using bags-of-words and TF-IDF metrics. In the context of insurance complaint handling, Dong and Wang [17] make use of synonyms and Chi-square statistics to reduce dimensionality.

Dealing with complaints as a multi-label classification problem can be effective, even when the original problem is not, due to the noisy nature of user-generated content. Ranking algorithms [10, 12] are a promising approach in this regard, providing a set of predictions sorted by confidence. These techniques have been applied in complaint analysis [5], although with modest results.

Kalyoncu et al. [9] approach customer complaint analysis from a topic modeling perspective, using techniques such as Latent Dirichlet Allocation (LDA) [2]. This work is not so much focused on automatically processing complaints, but instead on providing a visualization tool for mobile network operators.

3 Data

The dataset under study has been provided by ASAE. It contains a total of 48,850 complaints received by this governmental entity between 2014 and 2018, submitted by citizens, economic operators, public organizations or other organizations either by email or through a contact form in an official website. Each complaint contains its textual content and is classified with a single economic activity. This is the focus of this work, i.e., to train a classifier that is able to predict this activity (or a generalization thereof).

The economic activity taxonomy used by ASAE is hierarchical in nature. The first level contains 11 classes, and its imbalanced distribution is shown in Table 1. Generally, each class is composed of a number of sub-classes, which have a further decomposition level. Given the large number of second and third-level classes, we decided to train our classifiers to predict first-level classes only.

Since our goal is to aid ASAE staff in handling complaints, we have decided to base our classifications on their textual contents alone. The average complaint is 1,664 characters long after removing HTML tags and other artifacts, containing information on its subject matter, the targeted economic agent and contact information of the claimant.

4 Data Preprocessing and Feature Extraction

We have gone through a typical preprocessing pipeline, including tokenization and lemmatization. Based on [13], we have chosen to use NLTK[4], StanfordNLP[5]

[4] https://www.nltk.org/
[5] https://stanfordnlp.github.io/stanfordnlp/

Table 1. Distribution per classes

Class	# examples	# 2nd level subclasses
I - Primary Production	134	7
II - Industry	2031	26
III - Restoration and beverages	20899	4
IV - Wholesalers	299	4
V - Retail	5951	23
VI - Direct selling establishments	1	1
VII - Distance selling (by Catalog and Internet)	2856	1
VIII - Production and Trade	3335	69
IX - Service Providers	9933	85
X - Safety and Environment	696	62
Z - No activity identified	2715	*N/A*

and spaCy[6]. Given the lack of conclusive data on their performance for Portuguese, the non-exhaustive experiments shown in Table 2 were performed to analyze which were better to identify the economic activity of a complaint.

StanfordNLP was chosen for most experiments, given its competitive contribution to the task and because it is able to identify punctuation marks. Additionally, StanfordNLP provides specific and complete support for Portuguese and presents the data using the CoNLL-U format [16], which increases interoperability with other tools. After obtaining the lemmas, we remove punctuation marks and stop words (using NLTK's stop word list for Portuguese) before performing TF-IDF counts. Given that we have a single example for class VI, as per Table 1, we decided to leave it out of our classification problem.

To perform feature extraction, different data representation techniques were used: count, hashing and TF-IDF, as provided by scikit-learn [3]. The count technique transforms a collection of texts into a matrix of token counts. Hashing obtains a matrix of either token counts or binary occurrences, depending if we want counts or one-hot encoding. We used it to obtain token counts and compare the difference with the count technique because it has a few advantages, like low memory scalability. TF-IDF obtains features representing the importance of each token in the collection of all documents. For these three techniques, we present results obtained by using bags-of-words of 1-grams, 2-grams, 3-grams and intervals of 1 to 2-grams, 1 to 3-grams and 2 to 3-grams.

5 Predicting Economic Activity

The classification task addressed in this paper concerns predicting the economic activity targeted in a complaint. We focus on the first level of the hierarchy, as explained in Section 3. In order to find out which classifiers would allow us to obtain the best results, we decided to use Random Forests, Bernoulli NB, Multinomial NB, Complement NB, k-Nearest Neighbors, SVM, Decision Tree,

[6] https://spacy.io/

Extra Tree and Random (stratified). The latter will be used as baseline. All of them were implemented using scikit-learn[7] and the default parameters are used (for version 0.22), except those explicitly stated.

To split the original dataset into training and test set, we use 30% of the data for testing, while keeping the distribution of classes of the original dataset in both training and test sets. Following this procedure, we ensure that the trained classifier learns the real distribution of the data, and that the distribution is kept in the test set. Cross-validation was considered but given the considerable amount of training data it was deemed unnecessary to ensure consistency. This is important not only to ensure proper training but also to ensure that, when applying over/under sampling, no over/underfitting occurs in a class.

Our main performance metric was the accuracy score instead of the average macro-F1 score. We aim to provide a list of classes sorted by confidence and it is not critical to correctly classify minority classes. As a baseline we used a stratified random classifier that yielded an accuracy of 0.2504.

Table 2. Economic Activity Multiclass Classification accuracy scores using different tokenizers/lemmatizers

Classifier	StanfordNLP (baseline)	NLTK	spaCy - pt_core_news_sm	spaCy - xx_ent_wiki_sm
Random Forests	0.6787	0.6924	0.6818	0.6911
Bernoulli NB	0.5115	0.5363	0.5110	0.5185
Multinomial NB	0.4603	0.4719	0.4613	0.4648
Complement NB	0.5914	0.6263	0.5965	0.6066
K-Neighbors	0.6283	0.3146	0.6328	0.6180
SVM (linear)	**0.8075**	**0.8164**	**0.8093**	**0.8135**
Decision Tree	0.6659	0.6698	0.6669	0.6703
Extra Tree	0.5056	0.5228	0.5185	0.5166

In Table 2 we present the accuracy scores obtained using different tokenizers and lemmatizers to preprocess the text of the examples in the dataset. For this experiment, we used 1-gram TF-IDF to represent the features extracted. NLTK obtains the best scores overall, followed by spaCy and, finally, StanfordNLP. Nevertheless, we chose to continue using StanfordNLP because the performance loss is negligible and it provides PoS information, including punctuation marks. This proved useful to remove punctuation on all experiments and also experiment with removing adjectives. Furthermore, it has the advantage of having specific support for several languages, several more than the ones supported by NLTK and spaCy (although for now we are focusing on Portuguese).

Table 3 presents accuracy scores obtained using the different feature representation techniques discussed in Section 4. We used StanfordNLP for preprocessing and represent only 1-grams. Accuracy scores vary considerably depending on the

[7] https://scikit-learn.org/stable/

Table 3. Economic Activity Multiclass Classification accuracy scores using different feature extraction techniques

Classifier	Count	Hashing	TF-IDF
Random Forests	0.6958	0.6561	0.6787
Bernoulli NB	0.5115	0.4415	0.5115
Multinomial NB	0.6329	Error[1]	0.4603
Complement NB	0.6790	Error[1]	0.5914
K-Neighbors	0.5359	0.5750	0.6283
SVM (linear)	**0.7784**[2]	**0.7953**	**0.8075**
Decision Tree	0.6786	0.6671	0.6659
Extra Tree	0.4968	0.4865	0.5056

[1] Hashing may generate negative feature values, not supported by some classifiers.
[2] Failed to converge after 1,000 iterations.

classifier used, the best being obtained using SVM and TF-IDF. For that reason, subsequent experiments make use of TF-IDF.

Table 4. Economic Activity Multiclass Classification accuracy scores using different n-grams

Classifier	1-gram	1 to 2-grams	2-grams	1 to 3-grams	2 to 3-grams	3-grams
Random Forests	0.6737	0.6503	0.6323	0.6230	0.6127	0.5663
Bernoulli NB	0.5115	0.4763	0.4703	0.4622	0.4561	0.4495
Multinomial NB	0.4603	0.4568	0.4700	0.4568	0.4683	0.4733
Complement NB	0.5914	0.5432	0.5978	0.5381	0.5922	0.6320
K-Neighbors	0.6283	0.6152	0.5821	0.5950	0.5631	0.5413
SVM (linear)	**0.8075**	**0.8098**	**0.7640**	**0.8004**	**0.7396**	**0.6532**
Decision Tree	0.6659	0.6729	0.6121	0.6717	0.6120	0.5413
Extra Tree	0.5056	0.5338	0.5462	0.5541	0.5398	0.5298

In Table 4 we present the accuracy scores obtained using different n-grams when performing feature extraction with TF-IDF. It is not possible to conclude which is the best interval of n-grams because it depends on the classifier, but, for SVM, 1 to 2-grams is the best choice, followed by 1-gram. Because the difference between 1-grams and 1 to 2-grams in small for SVM, but higher for Random Forests, the following experiments use only 1-grams.

Taking into account the potential usage of the classifier, which is meant to help humans on analyzing complaints by providing likely classification labels (as opposed to imposing a definitive one), we looked at the performance of the classifier considering the ranking provided. In Table 5 we present accuracy scores obtained by accepting the 1st, 2nd and 3rd best probabilities. The second column shows the accuracy scores accepting as correct only the option with the highest probability. The third/fourth column shows the accuracy scores when accepting as correct one of the two/three options with the highest probabilities. For most

Table 5. Economic Activity Multiclass Classification accuracy scores for top-k predictions. Acc@k: accuracy scores considering the top-k (Acc@k) predicted classes, according to the confidence of the classifier predictions

Classifier	Acc@1	Acc@2	Acc@3
Random Forests	0.6787	0.8322	0.8885
Bernoulli NB	0.5115	0.7533	0.7913
Multinomial NB	0.4603	0.6790	0.7936
Complement NB	0.5914	0.8214	0.8873
K-Neighbors	0.6283	0.7699	0.8447
SVM (linear)	**0.8075**	**0.9031**	**0.9474**
Decision Tree	0.6659	0.7086	0.7226
Extra Tree	0.5056	0.5627	0.5703

classifiers, the accuracy of the top-2 is considerably higher than the accuracy considering the top-1. The 0.9474 score with SVM and top-3 demonstrates that presenting a set of classes sorted by confidence will be an effective help.

5.1 Feature Selection

We noticed that TF-IDF using 1-gram extracted 252,000 features, while only 101,159 are of interest when analyzing feature importance with Random Forests. As such, although a lot of features are extracted, a considerable part will be of no use to a classifier. For that reason, we explored feature selection via Latent Dirichlet Allocation (LDA), with the aim of bringing the number of features down while improving classification and training speed by clustering the features that are more important for the classification problem. However, as shown in Table 6, the use of LDA largely reduces the effectiveness of the classifiers. Moreover, although Random Forests presents an increase of 6% when raising the number of LDA components, most other classifiers maintain or even decrease accuracy scores. For this experiment, we used StanfordNLP and TF-IDF, extracting only 1-grams and analyzing top-1 predictions.

Table 6. Economic Activity Multiclass Classification accuracy scores using LDA

Classifier	No LDA (baseline)	10 components	100 components
Random Forests	0.6787	0.4053	**0.4612**
Bernoulli NB	0.5115	0.4278	0.4278
Multinomial NB	0.4603	-	0.4306
Complement NB	0.5914	0.4157	0.3561
K-Neighbors	0.6283	0.3995	0.4146
SVM (linear)	**0.8075**	**0.4484**	0.4424
Decision Tree	0.6659	0.3320	0.3525
Extra Tree	0.5056	0.3300	0.3419

Based on these results, we concluded that performing LDA is not effective for this classification task. Applying Principal Component Analysis (PCA) [18] has led to a similar result.

Finally, we performed experiments using the Recursive Feature Elimination and Cross-Validated selection (RFECV) approach [7]. This technique consists in training a classifier multiple times with different features and yielding the feature matrix that generated the best classifier according to a chosen metric. RFECV was tested with Complement NB because it is fast to train, resulting in a classifier with significantly better accuracy. On the other hand, testing with SVM has shown that this classifier does not benefit from further optimization.

5.2 Over and Under Sampling

As shown in Table 1, the class distribution for our problem is very imbalanced. To improve the overall classification performance and, more specifically, the performance on minority classes, we explore two widely used techniques to deal with imbalanced datasets [8]: *random under sampling* and *random over sampling*.

We have chosen to use the "imblearn" Python package[8]. There were three alternatives to perform the over sampling: RandomOverSampler (ROS), SMOTE and ADASYN. ROS duplicates some of the examples of the classes, increasing the number of examples of all classes to the number of examples of the class with the highest number of examples, as indicated in the documentation of "imblearn". SMOTE generates new samples by interpolation, not distinguishing between easy and hard examples. ADASYN generates new samples by interpolation, focusing on generating samples based on the original samples which are incorrectly classified using a k-Nearest Neighbors classifier. Because we were testing several different classifiers, including a k-Nearest Neighbors classifier, we decided to use the RandomOverSampler to reduce bias in the results. For random under sampling, RandomUnderSampler (RUS) was chosen to be comparable to the RandomOverSampler. RandomUnderSampler randomly selects a subset of data for the targeted classes, reducing the number of examples of each class to the number of examples of the class with the smallest number of examples.

Table 7 presents the accuracy and average macro-F1 scores obtained by performing random over sampling and random under sampling on the dataset. For these experiments, we used StanfordNLP for preprocessing and TF-IDF to represent the features extracted. Only 1-grams were extracted and only the top-1 was analyzed. As shown in Table 7, when performing random over sampling the accuracy scores related to Naive Bayes increased significantly and the accuracy of Random Forests also increased, but for all others it decreased. A similar situation can be observed regarding the corresponding average macro-F1 score. This is demonstrative that repeating the same data in the classes with a lower number of examples does not help distinguishing the different classes (except for Naive Bayes) and indicates that the classifiers are not predicting mostly the more frequent classes due to their amount of examples.

[8] https://imbalanced-learn.readthedocs.io/en/stable/

Table 7. Accuracy scores and average macro-F1 score using over or under sampling

Classifier	Accuracy (baseline)	Accuracy ROS	Accuracy RUS	Avg macro-F1 (baseline)	Avg macro-F1 ROS	Avg macro-F1 RUS
Random Forests	0.6787	0.7137	0.4402	0.42	0.49	0.30
Bernoulli NB	0.5115	0.6477	0.4703	0.18	0.48	0.28
Multinomial NB	0.4603	0.7299	0.5223	0.09	0.56	0.37
Complement NB	0.5914	0.7130	0.5258	0.28	0.52	0.37
K-Neighbors	0.6283	0.5456	0.3959	0.46	0.46	0.29
SVM (linear)	**0.8075**	**0.7985**	**0.5555**	**0.63**	**0.62**	**0.43**
Decision Tree	0.6659	0.6294	0.3678	0.45	0.44	0.26
Extra Tree	0.5056	0.4942	0.2074	0.33	0.32	0.15

On the other hand, when performing random under sampling, only the accuracy scores related to Bernoulli NB and Multinomial NB increased, while for all the other classifiers it has decreased significantly. All average macro-F1 score are relatively low, but 6 of them decreased and 3 of them increased. This is demonstrative that reducing the amount of examples for the classes with a higher number of examples reduces the ability of distinguishing the different classes.

5.3 Additional Experiments

An experiment performed to analyze the impact of the removal of adjectives identified by StanfordNLP was performed to identify if they were important for the classification task. This experiment was interesting because strong adjectives are apparently important for the classification task, but other weaker adjectives should not be. Depending on the amount and type of adjectives present in the dataset, their removal could reduce the amount of features that are irrelevant for the problem. Comparing the accuracy scores of all classifiers with the accuracy scores obtained by not removing the adjectives (baseline), as is the case in Table 5, the percentage was always the same, differing only on the permillage. These results are indicative that adjectives are partially important for the classifiers, although most of them have a low or even null importance/coefficient.

Experiments performed to increase the accuracy of SVM (with linear kernel) generating different class weights and balanced class weights (hyperparameterization) [8] obtained accuracy and average macro-F1 scores close to the ones obtained using the default parameters: a maximum accuracy of 0.8096 with a macro-F1 score of 0.63. Also, the different kernels available for SVM (linear, poly, rbf, sigmoid, precomputed) were tested and it was found that the linear kernel is the best in terms of accuracy, immediately followed by the sigmoid kernel, and that the sigmoid kernel is the best in terms of average macro-F1 score, immediately followed by the linear kernel. Finally, experiments performed to test the use of ensembles based on decision trees, which usually have interesting performances, provided accuracy scores higher than the ones obtained using Random Forests, but considerably lower than the ones provided by SVM.

6 Error Analysis

Based on the different accuracy and average macro-F1 scores obtained, we decided to focus on SVM for the sake of error analysis. We show the obtained confusion matrix in Table 8, when considering top-1 classification only. The influence of the majority class III is visible, but also of the second majority class IX. Class Z, where there is no identified economic activity, seems to be the most ambiguous for the classifier.

Table 8. Confusion matrix of the baseline SVM (Top-1)

		Predicted									
		I	II	III	IV	V	VII	VIII	IX	X	Z
Actual	I	14	5	10	1	5	0	1	0	0	4
	II	1	324	155	2	61	2	8	26	0	30
	III	0	37	5935	1	72	7	30	160	2	24
	IV	0	9	16	22	22	0	3	7	0	11
	V	1	26	184	3	1454	16	32	42	1	26
	VII	0	0	16	0	7	722	26	62	1	23
	VIII	1	18	126	1	61	31	596	114	6	46
	IX	0	5	314	0	26	30	83	2479	10	33
	X	0	0	17	1	6	8	31	52	81	12
	Z	2	35	181	3	72	55	93	163	6	204

To better understand in which situations the classifier was making erroneous predictions, we randomly sampled 50 examples from the dataset where the classifier was not capable of correctly predicting (from the top-3 predictions) the gold-standard class. Based on a manual analysis of such cases, we were able to draw the following observations:

- The dataset includes some short text complaints, not providing enough information to classify their target economic activity. Furthermore, a small number of complaints are not written in Portuguese. Some complaint texts are followed by non complaint-related content, sometimes in English.[9]
- Some classes exhibit semantic overlap (to a certain degree), thus confusing the classifier. For example, class VIII apparently overlaps with classes II and V. Moreover, while being labeled with a given class, some complaints contain words that are highly related with a different class.
- A non-negligible number of examples refer to previously submitted complaints, either to provide more data or to request information on their status. These cases do not contain the complaint itself, the same happening when a short text simply includes meta-data or points to an attached file.
- Finally, we were able to identify some complaints that have been misclassified by the human operator.

[9] Complaints received by e-mail often include "think twice before printing" appeals.

7 Conclusions and Future Work

For the imbalanced complaints dataset of ASAE, SVM with a linear kernel proved to be the best option among the experimented models. It is reasonably fast, allows to get probability scores and gives the best accuracy scores and average macro-F1. It is particularly valuable if we need a ranked output, given its high accuracy when aggregating the top-3 predicted classes. It is interesting to note that removing punctuation and stop words after lemmatization, using TF-IDF and training the SVM generates better accuracy scores than using additional techniques like feature selection and different quantities of n-grams.

After analyzing misclassified examples, several improvements have been planned. Non-Portuguese complaints need to be ignored, as the number of examples is too low to warrant a multilingual classifier. Furthermore, we aim to further assess how to discard texts that are simply not informative enough to consider as valid complaints (besides empty complaints, which the system correctly classifies). We also aim to tackle additional classification problems exploring this rich dataset. The ideas presented in this work will be the baseline for these future classifiers. We intend to explore recent advances on word embeddings approaches and deep learning techniques, and compare the results obtained with the models presented in this paper. The end goal is to create a system that will greatly assist ASAE personnel when handling these complaints.

Acknowledgements

This work is supported by project IA.SAE, funded by Fundação para a Ciência e a Tecnologia (FCT) through program INCoDe.2030. Gil Rocha is supported by a PhD studentship (with reference SFRH/BD/140125/2018) from FCT.

References

1. Batrinca, B., Treleaven, P.C.: Social media analytics: a survey of techniques, tools and platforms. AI & Society **30**(1), 89–116 (Feb 2015)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**, 993–1022 (Mar 2003)
3. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108–122 (2013)
4. Diaz, G.O., Ng, V.: Modeling and prediction of online product review helpfulness: A survey. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. pp. 698–708 (2018)
5. Fauzan, A., Khodra, M.L.: Automatic Multilabel Categorization using Learning to Rank Framework for Complaint Text on Bandung Government. In: 2014 Int. Conf. of Advanced Informatics: Concept, Theory and Application (ICAICTA). pp. 28–33. Institut Teknologi Bandung, IEEE (2014)

6. Forte, A.C., Brazdil, P.B.: Determining the level of clients' dissatisfaction from their commentaries. In: Computational Processing of the Portuguese Language - 12th Int. Conf., PROPOR 2016. Lecture Notes in Computer Science, vol. 9727, pp. 74–85. Springer (2016)
7. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46**(1), 389–422 (2002)
8. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. on Knowl. and Data Eng. **21**(9), 1263–1284 (Sep 2009). https://doi.org/10.1109/TKDE.2008.239, https://doi.org/10.1109/TKDE.2008.239
9. Kalyoncu, F., Zeydan, E., Yigit, I.O., Yildirim, A.: A Customer Complaint Analysis Tool for Mobile Network Operators. In: 2018 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM). pp. 609–612. IEEE (2018)
10. Li, H.: Learning to rank for information retrieval and natural language processing. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publ., San Rafael, CA, 2 edn. (2014)
11. Liu, C.H., Moriya, Y., Poncelas, A., Groves, D.: IJCNLP-2017 task 4: Customer feedback analysis. In: Proceedings of the IJCNLP 2017, Shared Tasks. pp. 26–33. Asian Federation of Natural Language Processing, Taipei, Taiwan (Dec 2017)
12. Momeni, E., Cardie, C., Diakopoulos, N.: A survey on assessment and ranking methodologies for user-generated content on the web. ACM Comput. Surv. **48**(3), 41:1–41:49 (Dec 2015)
13. Omran, F.N.A.A., Treude, C.: Choosing an nlp library for analyzing software documentation: A systematic literature review and a series of experiments. In: Proceedings of the 14th Int. Conf. on Mining Software Repositories. pp. 187–197. MSR '17, IEEE Press, Piscataway, NJ, USA (2017)
14. Ordenes, F.V., Theodoulidis, B., Burton, J., Gruber, T., Zaki, M.: Analyzing Customer Experience Feedback Using Text Mining: A Linguistics-Based Approach. Journal of Service Research **17**(3), 278–295 (2014)
15. Petz, G., Karpowicz, M., Fürschuß, H., Auinger, A., Stříteský, V., Holzinger, A.: Opinion mining on the web 2.0 – characteristics of user generated content and their impacts. In: Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data. pp. 35–46. Springer (2013)
16. Qi, P., Dozat, T., Zhang, Y., Manning, C.D.: Universal dependency parsing from scratch. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. pp. 160–170. Association for Computational Linguistics, Brussels, Belgium (October 2018)
17. Shuang Dong, Zhihong Wang: Evaluating service quality in insurance customer complaint handling throught text categorization. In: 2015 Int. Conf. on Logistics, Informatics and Service Sciences (LISS). pp. 1–5. IEEE (2015)
18. Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B **61**(3), 611–622 (1999)