CIA: Citizen Contact Center Agent Assistant

Luís Miguel Santos Neto

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. André Ferreira Ferrão Couto e Vasconcelos
Prof. Ricardo Daniel Santos Faro Marques Ribeiro

Examination Committee

Chairperson: Prof. Francisco João Duarte Cordeiro Correia dos Santos
Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Prof. Nuno João Neves Mamede

January 2021

Acknowledgments

This thesis is not just a final step toward academic graduation. It is proof that anything worth obtaining requires not only hard work but strength of will and support from the people that believe in us. Thus I most sincerely want to express my gratitude to everyone that supported me during not only this thesis but also the entire academic process which this milestone concludes. A special mention needs to be made to:

Prof. André Vasconcelos, who invested me in this project and readily put his attention and dedication at my disposal to guide me through all the obstacles that I came across. From the start, his availability and advice were more than I could ever expect.

Prof. Ricardo Ribeiro, was the pillar of this journey. His availability to meet with me, to review and guide my work, to constructively criticize my decisions and his attention to every detail was relentless. Without him, I could not produce the work that I here present.

Daniel Andrade and Filipe Ganança from the contact center side, which were always ready to help me with my challenges and who provided me with valuable insights.

My girlfriend for being at my side in the good times and in the hard times, always believing in my success and always pushing me to do my best. She was always present to lift me up, to talk, to advise, and to balance work and joy.

And finally the most important people in my life: my family, which supported me during my entire academic path and without whom I would not be able to accomplish everything I did. My mother, especially, who is always looking out for me, always available to do everything beyond her reach to provide me with everything I could possibly need, always caring, and always at my side, I dedicate this work to her.

Lisboa, January 29, 2021
Luís Neto

Iacta alea est!
(The die has been cast!)

Julius Caesar

Abstract

The contact center for Portuguese public services brings together a unique dataset, which presents itself as a very valuable study object in the field of Natural Language Processing (NLP): a collection of emails exchanged between Portuguese citizens and the contact center, which are categorized on various subjects, organized in a hierarchy. Data extracted from a real-world context presents serious challenges, namely in terms of the data quality, the class balance, the structure of classes, and the class relationship. For the purpose of developing a hierarchical email classification system, we present a study on data quantity and data quality impact on classification performance, by performing experiences on different datasets sizes and applying strategies of manual and automatic noise cleaning and data normalization strategies. Furthermore, we propose several new approaches to class imbalance, label noise, and class overlap, in the context of this problem, taking advantage of the hierarchical class structure and class semantic relations to re-organize data and re-structure classes. Finally, an extensive set of Machine Learning and Deep Learning classification techniques are studied in conjunction with different strategies of data representation, features extraction and selection and the aforementioned reorganization of the hierarchical structure of classes. The best performing system combines an XGBoost classifier and TF-IDF in a level-based hierarchical approach. Furthermore we demonstrate an increase in performance proportional to the quality of the data and the structure of the classes.

Keywords

Hierarchical Classification; Data Noise; Class Imbalance; Text Classification; Category Hierarchy Re-structure; Machine Learning

Resumo

O centro de contactos dos serviços públicos portugueses reúne um conjunto de dados único, que se apresenta como um objeto de estudo muito valioso na área do Processamento de Linguagem Natural (PNL): uma coleção de emails trocados entre cidadãos portugueses e o centro de contactos, os quais são categorizados em vários assuntos, organizados numa hierarquia. Os dados extraídos de um contexto do mundo real apresentam sérios desafios, nomeadamente em termos da qualidade dos dados, o equilíbrio das classes, a estrutura das classes e a relação das classes. Com o objetivo de desenvolver um sistema de classificação hierárquica de e-mails, apresentamos um estudo sobre o impacto da quantidade de dados e da qualidade dos dados no desempenho da classificação, realizando experiências em diferentes tamanhos de conjuntos de dados e aplicando estratégias manuais e automáticas de limpeza de ruído e de normalização de dados. Além disso, propomos várias novas abordagens para o desequilíbrio de classes, ruído de anotações e sobreposição de classes, no contexto deste problema, aproveitando a estrutura hierárquica de classes e as relações semânticas de classe para reorganizar os dados e reestruturar as classes. Finalmente, um extenso conjunto de algoritmos de Aprendizagem Automática e Aprendizagem Profunda são estudados em conjunto com diferentes estratégias de representação de dados, extração e seleção de features e a reorganização da estrutura hierárquica de classes. O sistema com melhor desempenho combina os algoritmos XGBoost e TF-IDF numa abordagem hierárquica baseada em nível. Além disso, demonstramos um aumento de desempenho proporcional à qualidade dos dados e à estrutura das classes.

Palavras Chave

Classificação Hierárquica; Ruído de Dados; Desequilíbrio de Classes; Classificação de Texto; Reestruturação de Hierarquias de Classificação; Aprendizagem Automática

Contents

List of Figures

List of Tables

Listings

Acronyms

TF-IDF	Term Frequency - Inverse Document Frequency
NLP	Natural Language Processing
DL	Deep Learning
ML	Machine Learning
DM	Data Mining
AI	Artificial Intelligence
RPA	Robotic Process Automation
CRISP-DM	CRoss Industry Standard Process for Data Mining
BDA	Big Data Analytics
NL	Natural Language
TC	Text Classification
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
MNB	Multinomial Naive Bayes
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
NB	Naïve Bayes
MLP	Multi-Layer Perceptron
IR	Information Retrieval
BOW	Bag of Words
WE	Word Embedding
XGBoost	eXtreme Gradient Boost
LSTM	Long Short Term Memory unit

Bi-LSTM	Bidirectional Long Short Term Memory unit
BERT	Bidirectional Encoder Representations from Transformers
OOV	Out Of Vocabulary
NLTK	Natural Language ToolKit
TruncatedSVD	Truncated Singular Value Decomposition
VSM	Vector Space Model
TF	Term-Frequency
SFS	Semantic Feature Selection
DT	Decision Tree
IG	Information Gain
NN	Neural Networks
LLSF	Linear Least-squares Fit
NCAR	Noise Completely at Random
NAR	Noise at Random
NNAR	Noise Not at Random
IDF	Inverse Document Frequency
LDA	Latent Dirichlet Allocation
NER	Named Entity Recognition
OCR	Optical Character Recognition
MSVM-KNN	Multi-class Support Vector Machine (SVM)-K-Nearest Neighbors (KNN)
ALC	Automatic Literature Categorization
HDLTex	Hierarchical Deep Learning Text Classifier
DNN	Deep Neural Networks
Bi-LSTM	Bidirectional Long Short Term Memory
WOS	Web of Science
HDL	Hierarchical Deep Learning
LR	Logistic Regression
BoW	Bag of Words
RF	Random Forest

JSON	JavaScript Object Notation
CSV	Comma-Separated Values
MI	Mutual Information
RMDL	Random Model Deep Learning
MNB	Multinomial Naïve Bayes
SVC	Support Vector Classifier
SGD	Stochastic Gradient Descent
OvO	One vs One
OvR	One vs Rest (One vs All)
POS	Part-of-Speech
ACM	Assossiation for Computing Machinery
SIGKDD	Special Interest Group on Knowledge Discovery and Data
DAG	Directed Acyclic Graph

1

Introduction

The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency

– Bill Gates

Contents

1.1 Motivation

Since it was introduced to the world, email has become increasingly vital to our daily life. It is not only an essential tool for personal communication, but also for personal management of information, events, tasks and activities.

As a result, email has become a key tool for interpersonal communication at the workplace, and therefore for many people a majority of their workday is spent within email. Team organization, project management, information exchange, decision making and client support are only a few of a company's daily processes where email is crucial. With loads and loads of incoming messages (some important, some junk), handling emails has become a tedious task. Such overload popped the need for email automation, which is being mentioned in the last two decades as an essential target for Machine Learning (ML), Data Mining (DM) and Natural Language Processing (NLP) disciplines [Whittaker and Sidner, 1997,Brutlag and Meek, 2000,The Radicati Group, 2015,Mujtaba et al., 2017]. Recent studies [Mujtaba et al., 2017, Park et al., 2019] find that current email automation opportunities fall under seven macro categories: Triage and Prioritization, Inbox Management, Email Categorization, Email Sending Automation, Email Modes, Inbox Presentation, and Mass Aggregation and Processing of Emails. Most of these categories describe tasks and processes in which automation solutions rely on the subject field of NLP.

NLP is a field study of Artificial Intelligence (AI) that deals with the interaction between computers and humans using the natural language. The ultimate goal is to ultimately make sense of the human languages in a manner that is valuable, by reading, understanding, writing, etc. Most NLP techniques rely on ML to derive meaning from human languages. It is nowadays one of the most rapidly growing fields of study and most of the breakthroughs in NLP in recent years are supported by the use of Deep Learning (DL). The popularity, fast-paced development and current rate of breakthroughs in the domain of NLP create the perfect environment to venture on the development of solutions to answer the aforementioned opportunities in email automation.

1.2 Context

The contact center of the Portuguese public services is a central channel of communication and interaction between the citizens or enterprises and the Portuguese public services[1][2]. The communication channel between the citizen and the contact center is operated over the exchange of emails and phone calls. Despite phone calls being the main channel, emails make about *1/5* of the throughput (an average of 500 daily incoming emails)[3] and this amount is predicted only to increase. Despite being subject to

[1] https://www.ama.gov.pt/web/agencia-para-a-modernizacao-administrativa/inicio

[2] https://eportugal.gov.pt/

[3] Data from the fourth quarter of 2019

a well defined process of triage, prioritization, categorization and answer, the amount of emails that are received everyday is too big and results in email overload. Naturally, of the aforementioned categories of email automation opportunities, Triage and Prioritization, and Email Categorization are the ones that characterise this scenario.

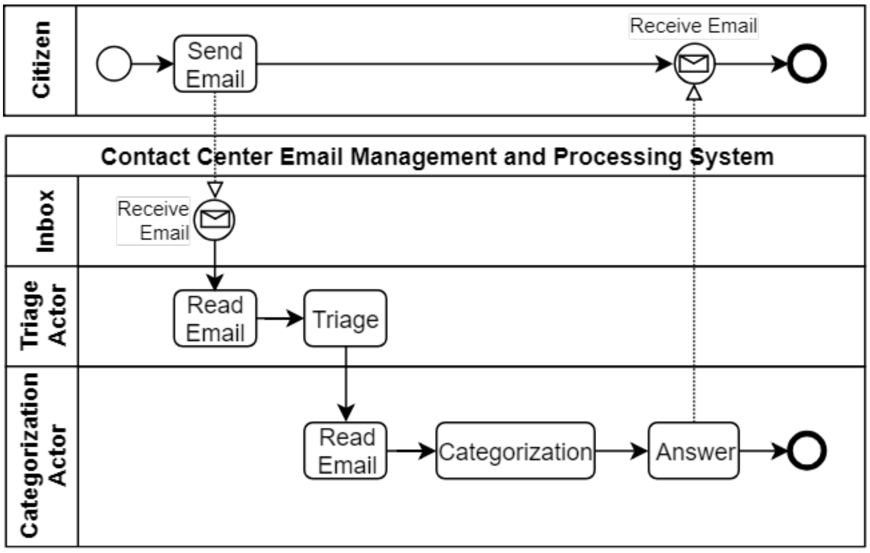


Figure 1.1: Contact Center Email Processing Workflow

Figure 1.1 represents the flow of tasks regarding emails management and processing at the contact center. For being simple, well-defined and, above all, repetitive, this process is an ideal candidate for Robotic Process Automation (RPA). The idea behind RPA is to take the repetitive workload and automate it using RPA bots so that the employees could focus on more value adding tasks and decision making to the organization. The RPA bot would also help to reduce the human errors and make processes more efficient, which results in cost saving and productivity increase [Van der Aalst et al., 2018].

To automate the process of triage, prioritization and categorization of email, a system must be developed that for each email received automatically outputs a priority and category label: an email classification system. Despite the popularity of the NLP field, the persistent problem of email overload and the fast-paced development of new tools and applications that rely on NLP to answer a diversity of challenges and problems, not only there is just a small set of solutions regarding email classification (compared to other types of text classification domains), but each problem needs a solution to be developed and adapted considering the domain of the problem.

The challenges that arise from the development of an email classification system are numerous and intricate. This thesis focuses on identifying these challenges, studying the possible approaches to each one of them, developing the corresponding solutions and ultimately creating a successful email classification system adapted to the domain of the Contact Center of the Portuguese Public Services.

1.3 Objectives

The main objective of this thesis is to **develop a system to classify emails**. Under the scenario presented in the previous section, the objective will be, more specifically, to develop an email classification system able to classify emails on the domain of the Contact Center of the Portuguese Public Services, and by doing so, to reduce the human effort on the tasks of emails triage, prioritization and classification or even to completely replace it.

To build a classification system, classifiers need to be trained, and to train a classifier, data is needed. This creates the challenges to:

1. Collect data from the business context and compile it into a dataset;
2. Explore the data compiled to identify the problems it may have and characterise it;
3. Develop solutions to deal with data problems;
4. Review the approaches to feature selection and feature extraction;
5. Survey the state of the art regarding classification strategies;
6. Develop approaches to automatically classify data;
7. Compare the results obtained by the chosen approaches;
8. Choose and refine the best approaches to obtain the best performance.

Additionally, given the nature of the domain of the problem, there are several specifications that need to be highlighted, namely the fact that it is a scenario in the real world with real data that presents challenges scarcely studied in literature which deals mostly with artificial and carefully curated datasets.

The fact that the data is written in Portuguese (which is far from the standard of the text classification systems literature) adds to the difficulty of the objectives.

1.4 Methodology

The methodology chosen to support this work was CRoss Industry Standard Process for Data Mining (CRISP-DM). The CRISP-DM is a widely used and popular methodology for analytics, data mining and data science projects [Piatetsky, 2014]. It is, in general, a set of guidelines to help plan, organize and execute a project, which fits the demands of this email classification system project.

Figure 1.2 provides an overview of the life cycle of a data science project. The outer circle symbolizes the cyclic nature of the process and the arrows inside the sequence, which allow to step back on some sequences of the process, mean the six phases are more agile than rigid.

The six high-level phases of CRISP-DM describe the flow of project:

1. **Business Understanding**: This initial phase focuses on understanding the project goals and requirements from a business perspective to give a context to the objectives. The main tasks are:

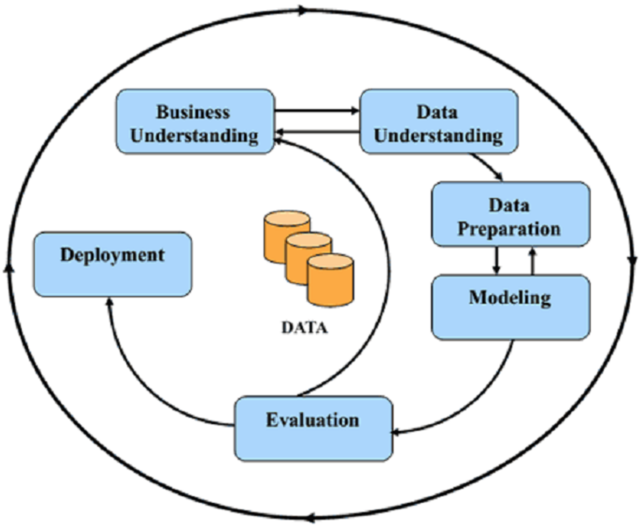


Figure 1.2: CRISP-DM Process Flow Diagram

- Understand the business background and determine the business objectives and success criteria;
- Assess the situation (resources available, requirements, constraints, risks, contingencies, terminology, costs and benefits);
- Draw a project plan, define project goals and project success criteria.

2. **Data Understanding**: This phase is all about building a data set. The main tasks are:
   - Collect, explore and describe data;
   - Verify data quality.
3. **Data Preparation**: The data preparation phase involves all the activities needed to prepare the data for the modeling phase. The main tasks are:
   - Clean, format and select data
   - Derive data attributes
4. **Modeling**: There is a close connection between Modeling and Data Preparation because the quality of the output of this phase is closely intertwined with the output of the Data Preparation phase. It is what is traditionally referred as Data Mining (DM) [Azevedo and Santos, 2008]. In this phase various models are considered being the main tasks to:
   - Select and build a model;
   - Assess model and tune parameters to optimal values;
   - Generate test designs.
5. **Evaluation**: As the name suggests, in this phase there is an overall analysis of the process that resulted from the decisions taken in previous phases. The tasks at hand may be summed up to:

  - Compare the data mining results with the business success criteria;
  - Review the process that was executed;
  - Determine the next possible actions.

6. **Deployment**: The deployment phase may be as simple as a presentation of the results or complex such as the deployment of an email classification system. It usually involves:
   - Developing a deployment plan;
   - Producing a final report;
   - Reviewing/documenting the project/experience.

1.5 Contributions

Several contributions from this thesis may be named, but there is one that stands out: Automatic classification over classes organized in a hierarchical structure is still very understudied when compared to the common ”flat” classification approaches. This work proposes strategies that take advantage of the hierarchical nature of classes by manipulating their relations to counter the well known problem of class imbalance and at the same time counter other issues such as class overlap and label noise.

Additionally, since the text data used in this project has the particularity of having been written by an extremely large and diversified number of sources and annotated by a group of professionals, the reports over data quality and the measures taken to increase it stand as an important contribution.

These and other contributions may be summarized as:

- in data collection and preprocessing:
  - An analysis on the challenges in transforming raw data into a dataset;
  - A comprehensive overview over the quality of data and how its aspects influence the performance of the tasks to be executed.
- in feature extraction/selection:
  - An overview of the most popular features extraction and selection methods;
  - A comparative review over the application of such methods;
  - A report on how feature engineering affects classification performance.
- in text classification:
  - An analysis on the application of several text classification approaches over the dataset;
  - A review over the results obtained by the strategies adopted, regarding feature extraction and selection and class structure redesign approaches.

1.6 Document Outline

The work that follows is structured in six chapters. Chapter 2 contextualizes the work in the email overload/automation scenario and introduces the fundamental concepts that will be used throughout this document. Chapter 3 analyses works in three scopes: works in similar contexts or with similar goals, works with similar data and works that use similar strategies, in comparison to the one being presented in this thesis. Chapter 4 characterizes data in its context and by its quality and presents the evolution of the solutions developed in order to refine it. Chapter 5 starts with a graphic description of the workflow involved in the training of our data classification system and introduces all the algorithms that were used during the development, namely the algorithms that counter the major problems of our data. It introduces the evaluation metrics used to measure our solution's performance and concludes with an overview of the development process in order to substantiate the decisions made during development. Chapter 6 illustrates the different solutions developed in search of the optimal performance, comparing the effects on data and with the variations of performance. It is completed by showing the performance behaviour with different algorithms at several steps of the classification system. The last chapter (Chapter 7) finalizes this thesis by reviewing the work described in comparison with the propositions initially declared, checks the contributions made, the limitations endured and concludes by outlining the future work in the development of this project and in the scope of the research fields that are the ground of this thesis.

2

Background and Fundamental Concepts

Automation is good, so long as you know exactly where to put the machine.

– Eliyahu M. Goldratt

Contents

2.1 Data Processing

The first two steps of our methodology are Business Understanding and Data Understanding. Figure 1.2 shows two arrows connecting these steps in both directions because they are closely related and work in an exclusive cycle. One of the tasks in the business understanding phase is to understand how data flows and what it means in the context of the business processes and in data understanding to collect it, explore it and describe it. There are several aspects to consider when managing data and in this work there are three that play a crucial role for different reasons: information sensitivity is important when collecting/assessing data, data type is important when handling/manipulating data and data quality is important for the performance of the processes that are applied to or operate with data.

"Information sensitivity is the control of access to information or knowledge that might result in loss of an advantage or level of security if disclosed to others"[1]. Data sensitivity concerns information that should be protected from unauthorized access or disclosure due to its sensitive nature. This might include personal information, such as social security numbers, id card numbers, telephone numbers, birthdays and names, business information, such as salaries, business proprietary information, etc. Before being disclosed to a third-party (for engineering experiments such as the one that grounds this work for example), sensitive data usually goes through a data anonymization process where specific data is altered by the application of some data anonymization techniques, in a way where the sensitive information is not disclosed. The most common data anonymization techniques are the following:

1. Data masking: hiding data with altered values. For example replacing a value with '*'.
2. Pseudonymization: replaces private identifiers with fake identifiers or pseudonyms. For example replacing a girls name with just 'Maria'.
3. Generalization: deliberately removes some of the data to make it less identifiable. For example removing all social security numbers from a dataset.
4. Data swapping/Shuffling/Permutation: rearrange values. For example, mixing the order of words in a text.
5. Data perturbation: adds random noise to data: For example multiplying all numbers by '5'.
6. Synthetic data: produces fake information that has no connection to real events.

The type of data influences the course of data-related experiments, such as this work, and in that matter it is important to distinguish between three types:

1. Structured data: Generalization or aggregation of items described by elementary attributes defined within a domain (range of possible values). Relational tables and statistical data are examples of common structured data.
2. Unstructured data: Generic sequence of symbols, typically in the form of natural language. Free

[1] https://en.wikipedia.org/wiki/Information_sensitivity

text such as news article or an email are the most common example.

3. Semi-structured data: Data that have a structure which has some degree of flexibility, such as JSON, XML or other markup language that has fields describing data, but data may be represented in multiple ways in multiple fields.

Data quality is defined as the wellness and appropriateness of data for use, to meet the requirements of some process or to meet user needs [Alizamini et al., 2010, Salih et al., 2019]. Literature usually mentions data quality in the context of Big Data Analytics (BDA). There is no global consensus on the definition of BDA but we will go with the definition used in [Favaretto et al., 2019]: ”ability to extract information from data using various techniques, such as artificial intelligence, mathematics, statistics and other techniques to support the decision making process”.

Normally, the most studied characteristic that impacts data quality, that is, its fitness for use or to meet the requirements of a process such as automatic classification, is data noise [Labadie and Prince, 2008]. Data noise is, in text data, the presence of terms that are out of the vocabulary of the domain, of spelling errors, of random punctuation or non-standardized terms. Data noise may be classified as natural (when it is the product of human mistakes) or systematic (when it results from a process). It is dealt within a process called data cleaning.

Data cleaning is the process of identifying and removing errors and inconsistencies from data in order to improve data quality. It comprises tasks such as checking if data contains misspellings, foreign or wrong words, sequences of punctuation marks, unanticipated abbreviations, ill-formed sentences, missing values, etc, and removing them. All these elements may be the source of low data quality which negatively impacts learning algorithms, especially those relying on NLP techniques such as syntactic and semantic analysis [Labadie and Prince, 2008].

2.2 Dataset Classes

In a structured dataset, data instances are assigned with one (leading to a single-label classification problem) or more classes (leading to a multi-label classification problem) from a set of classes which may contain two (leading to a binary classification problem) or more classes (leading to a multi-class classification problem). **The subject of this work is a single-label multi-class classification problem.**

Classes may have no relationship (flat classes), have a ranking/ordering relationship (ordinal classes) or even be organized in a hierarchical structure (**hierarchical classes**). The classes relationship/structure has high influence on the approaches that may be employed during the classification process [Silla and Freitas, 2011]. Furthermore, not only the structure of the classes influences the classification process, but also class issues such as **class imbalance**, **class overlap** and **label noise** have huge impact.

2.2.1 Hierarchical Class Structure

In a classification problem, classes may be organized in one of three types of structure: flat structure, ordinal structure or hierarchical (tree) structure.

Typically in large-scale data environments, there are huge amounts of data which are organized in complicated structures such as hierarchies. A hierarchy employs a parent-child relationship among classes, meaning that a data instance belonging to a class also belongs to its ancestor class.

A hierarchical tree is defined as a pair ($C, \prec$) where C is the set of all classes and $\prec$ is the relationship "is-a", in this case, the "subclass-of" relationship. This "subclass-of" relationship has the following properties:

1. Asymetry: if $m \prec j$, then $j \nprec m$ for every $m, j \in C$
2. Anti-reflexivity: $m \nprec m$ for every $m \in C$
3. Transitivity: if $m \prec j$ and $j \prec n$, then $m \prec n$ for every $m, j, n \in C$

In a tree structure, the top node (has no parent node) is called root node, the nodes with no child are called leaf nodes, and the nodes with parent and child nodes are called inter-level nodes. Two or more nodes with the same parent node are called a sibling node.

The set of child classes of class m is defined by the parameter C_m and the number of documents of the child classes is defined as $|C_m|$. The parent category of class m is defined by the parameter p_m and the set of sibling categories of class m is defined by parameter S_m. h is the height of the tree structure.

2.2.2 Class Imbalance

The most common issue, class imbalance, is the situation where classes are not equally distributed among the data instances. In an imbalanced dataset, a class mapped to a high proportion of the data instances is called a majority class and a class mapped to a low proportion of the data instances is called a minority class. In this circumstances, the majority classes tend to bias the classifiers towards themselves, resulting in a poor classification of the minority classes. Furthermore, a multi-class scenario where there is one majority class and multiple minority classes is called a multi-minority problem, where the opposite is called a multi-majority problem.

To solve problems associated with class imbalance, various techniques have been proposed over the last three decades [Japkowicz, 2000, Ali et al., 2019, Esteves, 2020], however, there is still a lack of systematic and organized taxonomy. [Batista et al., 2004] studies the application of eight methods for balancing training data without implying a taxonomy of approaches, [Japkowicz, 2000] and [Haixiang et al., 2017] distinguish two kinds of methods, the former distinguishes "re-sampling" and "learning by recognition", and the latter "preprocessing strategies" and "cost-sensitive learning". [Arafat et al., 2019] and [Ali et al., 2019] distinguish three types of strategies: algorithm-level approaches, data-level approaches and cost-sensitive approaches. [Cruz et al., 2019] and [Galar et al., 2011] go further and distinguish a fourth strategy type: ensemble-based approaches. These four type of strategies are the following:

1. The **internal** or **algorithm-level** approaches modify the usual learning methods/algorithms to take the minority classes into account and not bias towards the majority classes.
2. **External** or **data-level** approaches include preprocessing of data in order to re-balance the distribution of classes and reduce the impact of the imbalanced classes on the classification performance.
3. **Cost-sensitive** strategies combine both internal and external approaches by assigning different costs to every class based on the distribution of data instances and by applying modified learning algorithms to these classes.
4. **Ensemble-based approaches** combine any of the previous strategies, namely external, with an ensemble-based algorithm.

2.2.3 Label Noise

Label Noise is a problem rooted at the dataset foundation. Generally, a dataset is a collection of data instances that were labelled, directly or indirectly by a human (it may have been labelled by a machine engineered by a human), considered to be a data domain expert. However humans make mistakes and machines may malfunction. These situations may be the source of data mislabeling, which damages the relation between the classes and the features [Frénay and Verleysen, 2013].

Label noise has been identified as one of the most problematic types of noise in the process of text classification [Roberts et al., 2010, Frénay and Verleysen, 2013]. Text classification consists in predicting the class of a text, using a model inferred from training data. Each data instance is associated with an observed/recorded label which corresponds to the true class of the data instance, but there may be a mismatch between the observed/recorded label and the true class of the data instance [Frénay and Kabán, 2014, Atkinson and Metsis, 2020]. A wrong association between the features of the data instance and the class are created by polluted labels from this mismatch process called label noise [Zhu and Wu, 2004]. The source of this type of noise, usually, lies in the hands of a domain expert who performs or checks data and may be caused by a design fault of the class structure, for example the definition of ambiguous classes, poor decisions, data entry errors, poor data quality, insufficient information, wrong observations, complex patterns, communication problems or data encoding [Roberts et al., 2010, Frénay and Verleysen, 2013, Frénay and Kabán, 2014, Müller and Markert, 2019, Atkinson and Metsis, 2020].

[Frénay and Kabán, 2014] propose a taxonomy inspired by the works of [Schafer and Graham, 2002] where they define three types of label noise:

- **Noise Completely at Random (NCAR)** occurs independently of the true class and of the values of the instance features (a wrong observation or data entry error for example).

- **Noise at Random (NAR)** where the probability of the mislabelling depends on the true class (for example, confusing two classes that are very similar);
- **Noise Not at Random (NNAR)** is the more general case, where the mislabelling depends on both the feature values and the true class (for example, having an object with some features typical of two similar classes and assigning the wrong one).

The main consequence of label noise is that it deteriorates the relation between the features and the true classes that is crucial for the classification process, resulting in a decrease of the prediction performance. Consequently, the complexity of inferred models as well as the number of required training instances may increase [Frénay and Verleysen, 2013]. In some domains, a systematic label noise may distort observed results in a degree which may cause or intensify class imbalance problems.

[Müller and Markert, 2019] organize existing strategies to deal with label noise in three groups. The first suggests designing robust algorithms that can cope with label noise, such as the ones presented by [Bootkrajang, 2016] and [Li et al., 2017]. The second and third rely on identifying mislabeled instances and removing or re-labelling them, respectively. These strategies usually rely on a two level approach to identify mislabeled data: in a very straightforward definition, a classifier is trained on data in order to identify mislabelled instances which are then presented to a human checker, as possible mislabelled instances. [Frénay and Verleysen, 2013] mention that bagging algorithms are more robust to label noise than boosting algorithms (such as AdaBoost); that the node splitting criterion of decision trees can improve label noise robustness; and, conclude that robust methods rely on outfitting avoidance to handle label noise.

Although the consequences of label noise (such as difficulty to identify relevant features, increased complexity of learning models, misrepresentation of class proportions and general decrease in classification performance,) are important and diverse, works focused on label noise are few, compared to feature noise. Most works use either synthetic datasets or real-world datasets added with synthetic noise [Müller and Markert, 2019, Atkinson and Metsis, 2020], datasets not widely known benchmarks and there are no standard approaches to deal with label noise [Frénay and Kabán, 2014].

2.2.4 Class Overlap

Since most literature revolves around binary classification problems, where the two classes are generally antipodes, few works address class overlap. However, in multi-class problems, specially where there is a large set of classes, the case where data instances with very similar characteristics belong to different classes is common. These instances are called overlapping data instances because they are located in the "overlapping region" of the feature space. This "overlapping region" is a region in the feature space where learning algorithms struggle to define a class boundary that results in the best performance and where most mislabelled instances are usually located [Lavanya et al., 2014, Xiong et al., 2010]. This problem is called class overlap and is mainly caused by two factors: label noise (as mentioned in the previous section) and class structure/schema issues [Zhuge and He, 2017].

Solutions to class overlap problem are usually based on the core idea of discarding, merging or separating the data instances located in the overlap region [Xiong et al., 2010]. Discarding implies losing data (the data instances in the overlap region), merging implies altering the class structure/schema (merging two classes into one, or creating a new class for the overlap region) and separating implies complex learning strategies that perform well in the overlap region.

2.2.5 Class Hierarchy Structure Issues

“Designing a hierarchy is hard because it is not always possible to anticipate how a hierarchy will be used by an application” [Snelting and Tip, 1998]. A hierarchical class structure normally represents a human abstraction, where the categories closer to the root are more general and gradually become more specialized following the paths from the root to the leaves. These structures are made to continuously accommodate resources that are constantly coming and that may change in pattern and in volume [Zhuge and He, 2017].

“The usefulness of a hierarchy heavily relies on the effectiveness of the hierarchy in properly organizing the existing data, and more importantly accommodating the newly available data into the hierarchy” [Yuan et al., 2012]. Research articles focus mainly on hierarchical classification under a predefined hierarchy. Hierarchies represent a view over the organization/distribution of data, but data evolves and changes in time, making a predefined hierarchy no longer able to correctly categorize new data. Hierarchies need to evolve, but they evolve at a much slower pace than data, and during this process two major problems may arise [Yuan et al., 2012, Zhuge and He, 2017]:

Structure irrelevance: a hierarchy may characterise the topic distribution of its data at an initial point, but as data evolves, there are no proper categories in the hierarchy to accommodate new data. As a result, some new data instances are labelled in less significant categories, data become less cohesive and some categories become less discriminative with respect to the current data distribution;

Semantics irrelevance: semantics change in time and a category name which used to represent a collection of data may become less adequate, with the data it represents becoming more semantically related with another category.

These problems negatively affect classification performance and sabotage the fundamental purpose of the hierarchical structure. They call for “categorical hierarchy maintenance” modifications in order to adapt the category to the new data reality (new patterns of data) and improve the classification performance [Yuan et al., 2012]. There are four basic cases where a hierarchy needs modifications:

1. Two categories under the same parent share too many common features to distinguish them clearly (class overlap);
2. A category belongs to more than one parent category;
3. Leaf categories become less cohesive with new data;
4. Parent category can no longer represent its child category.

Category hierarchy maintenance is not a trivial task and despite there have been little work on this subject, a variety of approaches have been suggested. Snelting and Tip, 1998 propose a manual approach to redesign a class hierarchy based on concept analysis [Wille, 1982] and conclude that the technique is capable of finding design anomalies such as class redundancies (which can be moved onto an upper class) and situations where a class may be split, thus able to restructure a hierarchy tree.

[Tang et al., 2006] propose a maintenance approach based on three operations (Promote, Merge and Demote) that changed relations between categories. Promote moves a category to the upper level, Demote moves a category to a lower level under one of its siblings, and Merge merges two siblings, by creating a 'super category' that abstracts both and puts them as child categories of this 'super category'. These approaches change only the relations of categories and not the categories directly.

[Yuan et al., 2012] propose an approach to modify a category with reference to an auxiliary hierarchy (a hierarchy covering the same topics (the same domain)) using three non-trivial operations (Sprout, Merge and Assign). Hidden topics may be discovered by comparing the semantics of both hierarchies. The Sprout operation relies on the following steps: for each category C, the data instances are projected on the auxiliary hierarchy, H_a. A set of categories from the auxiliary hierarchy H_a will contain a reasonable proportion of C's data instances which are said to represent C's hidden topics and can be 'sprouted' from C into child categories. The Merge operation aims to merge similar sibling categories, specially the 'sprouted' categories. The Assign operation assigns children categories from the auxiliary categories used to sprout a given category C as children categories of C.

[Zhuge and He, 2017] propose a maintenance approach that follows a two-phase (global and local) hierarchy adjustments. The global phase uses hierarchical clustering to generate a cluster tree that reflects the similarity between categories, to detect inappropriately located categories. The local phase detects topical changes by Latent Dirichlet Allocation (LDA) topic model and then adjusts the hierarchy with three local operations: Merge, Pull-Up and Split. Merge and Pull-Up are similar to [Tang et al., 2006]'s Merge and Promote operations, respectively. Split is the operation of creating two or more child categories from a given category, according to a Category Cohesion metric defined by [Tang et al., 2006].

[Tang et al., 2006] test their approach on three different datasets with semantic-based hierarchies, using a Multinomial Naïve Bayes classifier. They conclude that a semantically sound taxonomy may not necessarily lead to the best classification performance and that the taxonomy may be evolved and adapted to changes in data and lead to better classification performance. [Yuan et al., 2012] use their approach on three real-world hierarchies and Naïve Bayes and Support Vector Machine (SVM) classifiers, and obtain improvements in macro F1-measure score ranging between 30% and 60%. Furthermore, they performed a 'user study' to check if the modified hierarchies are topically cohesive and had a semantically meaningful structure and obtained positive results. [Zhuge and He, 2017] used hierarchical variants of benchmark datasets 20-Newsgroups, Reuters-21578, and DMOZ[2] dataset. Using SVM, they also obtained much better F-measure and accuracy scores on the modified hierarchies while improving structural balance and semantics.

2.2.6 Summary

Class structure issues, namely hierarchy structure problems, arise from poor design, from the evolution of data which it accommodates and from hierarchy maintenance, that is, the changes applied in the hierarchy to couple with the changes in data.

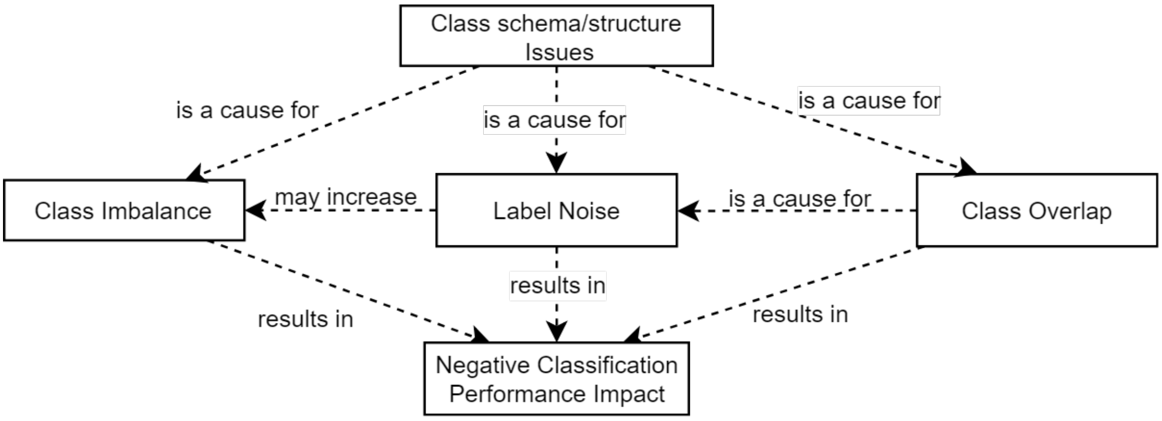


Figure 2.1: Class Issues Relationship

Poor hierarchy design influences class balance, because, among other issues, classes may be thought only for their semantic meaning and not distribute data correctly, leading to class imbalance. Furthermore, the evolution of data which it accommodates and the hierarchy maintenance, that is, the changes applied in the hierarchy to couple with the changes in data, that follows may increase the chance of *NAR* and *NNAR* label noise. Hierarchies are by nature more prone to class overlap because as the level depth increases, the boundary between classes decreases. The fact that hierarchy maintenance (for example the creation of new classes) only happens after the changes in data means that similar data will be present in more than one class, leading to class overlap. Class overlap can be easily related to NAR label noise, because of the class similarity issue. These concepts, described in sections 2.2.2, 2.2.4, 2.2.3 and 2.2.5, and their relation may be summarized by Figure 2.1.

[2] www.dmoz.org

2.3 Text Categorization

Text Categorization is one of the applications of Text Mining (the process responsible for identifying and extracting useful information from unstructured text) [Vijayarani et al., 2015] and is the process that deals with the assignment of predefined categories, topics, or labels to Natural Language (NL) texts or documents [Sebastiani, 2002].

The input of a Text Classification (TC) system is a set of documents (emails in the present work context) which represents the data to be classified. Traditionally, before the Classification step (which is the most important step in a TC system), there is the Document Representation step, during which a Feature Extraction process takes place, where documents unstructured data is cleaned and converted in a structured feature space. Then, since documents may contain a lot of features, a Feature Selection step may take place to reduce the dimensionality of the data and reduce the time and memory complexity of the classification step.

Fig. 2.2 show TC approaches divided in two types of classification: binary and multi-class. In the binary case there are only two categories of which one must be assigned to the document. This is the case of Spam Classification, which has been widely studied. In the multi-class case there is a set of categories and two possible cases: when a number of categories from 0 to C must be assigned to each document, it is called multi-label, and when only exactly one must be assigned it is called single-label [Sebastiani, 2002].

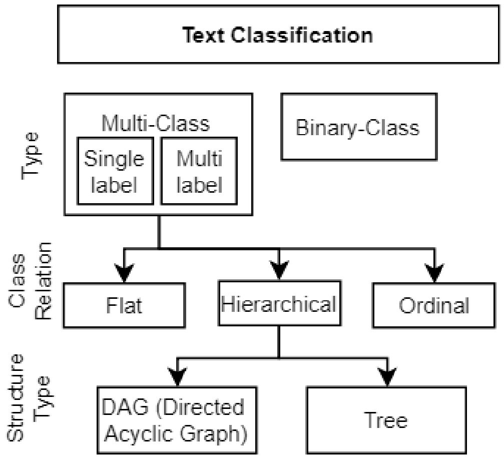


Figure 2.2: Text Classification Approaches Overview

Most Email Classification research falls under binary classification of emails, namely for spam detection, while multi-class classification is usually referred as ”email foldering” or ”multi-folder categorization”.

In the research results reported by Mujtaba et al., 2017 there is approximately 1 research article for email multi-class classification for every 7-8 research articles about email binary classification.

2.3.1 Class Relations

This work focus on **multi-class single-label classification of emails**. Furthermore, the set of classes used for email categorization show a hierarchical structure.

The **single-label hierarchical** classification problem may be defined as a 2-tuple (Υ, Φ) [Silla and Freitas, 2011], where

- Υ defines the type of structure that represents the hierarchical classes and their relationships. It may be *T* (Tree) or *Directed Acyclic Graph (DAG)*;
- Φ defines the label depth of the data instance. The instance may be labeled only with leaf node, *FD* (Full Depth), or with otherwise, *PD* (Partial Depth).

A **hierarchical classification** problem may deal with a flat or hierarchical classification approach where:

- A flat classification approach disregards the classes relations. This means the hierarchy is ignored and the classification approach predicts only the leaf nodes;
- The hierarchical classification approach may be further classified as Local or Global, where a local approach builds a classifier per level, per node or per parent-node and a global classifier learns a global single model for all classes.

An **ordinal classification** problem may deal with a flat or ordinal classification approach [Gutiérrez et al., 2016] where:

- A flat classification approach disregards the classes ordinal relations;
- The ordinal classification exploits the order information, by making use of ordinal regression approaches.

2.3.2 Classification Techniques

Email Categorization can be ultimately reduced to a process of Text Classification (for details see section A.1.3).

The most crucial step of a text classification pipeline is the classification step, where a classifier learns to link data instances with class labels. Choosing the best learning classifier is the subject of most studies regarding text classification.

Following Kowsari et al., 2019 survey on text classification algorithms the most popular methods for text classification may be grouped as traditional methods, ensemble methods and deep learning methods.

The most common traditional methods are Naïve Bayes (NB) classifiers, Decision Trees (DTs), rule-based classifiers, SVM classifiers, K-Nearest Neighbors (KNN) classifiers and related hybrid approaches. The most popular ensemble approaches are the decision tree-based method Random Forest and the boosting-based algorithms AdaBoost and eXtreme Gradient Boost (XGBoost). Common deep learning methods such as Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and other complex strategies capable of modelling complex data have also been adopted for text classification.

The NB classifier is based on Bayes theorem and has been widely adopted for document classification. Rule-based classifiers determine the class by applying rules that relate data patterns to classes. SVM is a binary method that consists on mapping input vectors into a high dimensional space and outputting the creation of a hyper-plane that separates data. KNN algorithm uses a distance measure, typically Euclidean distance to identify the k nearest neighbors of a test document among all documents in the training set, and scores the category candidates based on the class of k neighbors. Decision Trees are designed with the use of a hierarchical decomposition of the underlying data space with the use of different text features.

These algorithms are further explained and described in appendix C.4.

2.4 Information Retrieval

Information Retrieval (IR) is the process where some approaches are employed in order to extract the most meaningful and informative features of some source of (unstructured) data [Meadow et al., 2000] in order to satisfy some need. In text classification, information retrieval is usually mentioned as the processes of text pre-processing, dimensionality reduction, text representation, feature extraction and feature selection.

2.4.1 Pre-processing

One of the first and most common steps of text pre-processing is the removal of stop-words. Stop-words are frequent terms in a text which carry information of lesser importance. In the task of text classification, removing stop-words is usually a positive step, considering that stop-words provide little to no unique information and removing them increases the fraction of significant features and thus potentially allows accuracy to be increased. Removing stop-words also reduces the size of the dataset thus decreasing the time to train the model.

One of the crucial tasks used in IR, when applied to text, is the recognition of morphological variation and conceptual proximity of the words, called lexical normalization or word normalization. Two different normalization approaches are usually distinguished – stemming and lemmatization [Toman et al., 2006]. Both techniques produce a normalized form. Lemmatization produces a normalized form by replacing the suffix of a word with a different one or removing the suffix of a word completely to get the basic word form (lemma). Stemming combines the different forms of a word into a common normalized representation, called stem or radical.

When comparing both approaches, the results reported by the research community do not converge on a conclusion. For example [Balakrishnan and Lloyd-Yemoh, 2014] conclude, in their experiments, that lemmatization produced better precision compared to stemming but the difference of performance was insignificant, [Wibowo Haryanto et al., 2018] in an experiment comparing the influence of word normalization approaches in SVM text classification obtained better results using stemming than using lemmatization and [Toman et al., 2006] conclude in their comparison experiment that the best approach pre-processing approach is to omit word normalization.

2.4.2 Type of Representation

Information can be represented in multiple ways while keeping the same meaning. Text data is usually represented as numbers, since machines work better with this type of data. But within text data representation, there are different representations that fit different purposes. The best representations is the one that better fits the classifier model.

Text representations differ in the information they carry and have different impacts on the model. The most usual are one-hot encodings, Bag of Words (BOW) and Word Embedding (WE). They are further described in appendix C.1.

These text representations approaches are based on vectors that represent words or texts, which are modeled as elements of a vector space and their collection is said to be modelled as a Vector Space Model (VSM).

Usually traditional ML models use BOW text representations while DL models use WE word representations.

2.4.3 Dimensionality Reduction

Feature extraction is a process of dimensionality reduction popular in text categorization due to text high dimensionality. Feature extraction acts by selecting, combining or transforming variables into features, reducing the amount of data that must be processed, while still accurately describing the original data set. It generally reduces the amount of redundant data and promotes the speed of learning and generalization steps in the machine learning process.

Feature extraction is frequently reported in text classification processes, such as the one that grounds this work, with Term Frequency - Inverse Document Frequency (TF-IDF) being the most popular feature extraction algorithm. Some important feature extraction methods are further described in appendix C.2.

Feature selection is the process of automatically or manually selecting a subset of relevant features (those which contribute most to the model performance). Like feature extraction, feature selection is a dimensionality reduction approach designed to reduce the number of features, namely irrelevant or partially relevant features that can decrease the model performance, and to improve the performance by reducing the computational cost. Some of the most relevant feature selection algorithms are further described in appendix C.3.

3

Related Work

Humans are very good at making algorithms work eventually.

– Usama Fayyad, co-founder of KDD conferences and Assossiation for Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data (SIGKDD)

Contents

3.1 Motivation

Searching for problems similar to the one we have at hands should always be the first step for finding a solution. But, as previously mentioned, most email classification research is focused on binary classification problems (spam filtering and phishing filtering), leaving only some scarce email multi-class classification articles of relevance. Email multi-class classification systems, using unsupervised classification methods were not considered since we want to classify emails against a predefined set of categories and intent classification systems were also discarded because its problems/solutions have crucial differences from the ones at stake in this work. Since, there are not many published works regarding email single-label multi-class classification systems, the search was extended to similar problems such as IT help desk automated systems, customer support services systems and TC problems where data has similar characteristics to our data.

It is not only the classification that plays a huge role in this work, but due to the nature of data and the hierarchical structure of classes, there are issues regarding classes that have a big influence on the development of this project, being the source of important decisions and the cause of the results obtained.

3.2 Email Classification Systems

As early as 1996, W. Cohen notices a ”great deal of interest in systems that allow a technically naive user to easily construct a personalized system for filtering and classifying documents such as e-mail” [Cohen, 1996]. He explores how text classifiers, namely a ”keyword spotting rule” and a ”method based on TF-IDF weighting” perform on small datasets of hundreds of personal emails, over eleven categories, and realizes that both methods obtain significant generalizations, from a small number of examples, and comparable performance.

Neural Networks (NN) were applied by the first time (to the authors best knowledge) at email multi-class classification in 2003. [Clark et al., 2003] presented LINGER, ”a NN-based system for automatic e-mail classification”. LINGER used a MLP (MultiLayer Perceptron) trained with backpropagation, to classify five personal email datasets, containing 545, 423, 888, 926 and 982 emails into 7, 6, 11, 19 and 6 folders, respectively. The results obtained are proportional to the ratio between emails and folder with the best scores being (in terms of F1-measure) 79.44%, 43.10%, 76.81%, 39.92% and 83.18%, respectively. This experience indicates that the larger the amount of data in relation to the amount of the classes, the better the classification results. The experience also tested the influence of the feature selection method, using Information Gain (IG) and Variance. Performance results with one of the datasets showed a difference of 40.63% in terms of F1-measure, with the other datasets showing a difference of approximately 4 to 26%, suggesting that the feature selection method employed is of great relevance. The document representation methods used were BOW, TF-IDF, Term-Frequency (TF) and binary count.

In a very similar experience, [Yu and hua Zhu, 2009] proposed a new classification model, a modified MLP (to overcome the slow learning speed of the traditional MLP). This model was evaluated over four datasets with the smallest containing 1099 emails and the largest containing 4139 emails, distributed over eleven classes. One of the key factors of this experience is that the feature selection algorithm that was used was a Semantic Feature Selection (SFS), based on Truncated Singular Value Decomposition (TruncatedSVD) a method for reducing the dimensionality of large datasets and extracting the dominant features of the data.

Using the publicly available dataset of emails Enron [Klimt and Yang, 2004], [Xia et al., 2007] performed a comparative analysis on the performance between some-vs-rest round robin binarization method, a binary decision tree and SVM multiclass. Emails were represented by 'non stop-words' weighted by TF-IDF. The experimental results, evaluated on F1-measure score, showed that the binary decision tree is more effective than SVM and some-vs-rest round robin method binarization in email categorization and computationally less complex in training. The experience showed that SFS not only was able to drastically reduce the number of features but also improve accuracy and efficiency when compared to VSM.

Similar to Email Categorization, [Silva et al., 2018] in the IT incident management context, study the application of an automatic text classification system for incident tickets categorization. They study the application of several techniques at the preprocessing phase, namely tokenization, stemming, eliminating stop-words, named-entity recognition, and TF-IDF based document representation. The classifiers that were used were SVM and KNN. The data used contained tickets written in English and classes were distributed in a hierarchically, with the first level containing 10 categories and the second containing 94 categories in total. During the experiments, the datasets used had 2000 incidents per category. The work concluded that a short description of an incident lead to better accuracy score, that the choice of document representation method (TF-IDF, TF or Inverse Document Frequency (IDF)), the choice of tokenizer, the choice of whether to use stemming or not, and the choice of using Named Entity Recognition (NER) or not had little to no influence over the accuracy score. As expected, accuracy score was lower in the second level of the hierarchy, compared to the first-level.

Similarly to a contact center, the IT support help desk handles emails/messages regarding a set of subjects in a repetitive pattern. Aiming to automate this process, [Shanmugalingam et al., 2019] proposed an approach focused on automating repetitive task (RPA), in which they present a classification model to categorize requests (emails) directed to the help desk. Emails are categorized with one category from each of the three levels of categories, containing 84, 8 and 77 unique categories.

All emails undergo a pre-process stage were signatures, greetings and URLs are removed, key body content is extracted and analyzed using Microsoft LUIS [Williams et al., 2015] in order to distill the intent and attachments are analyzed using Optical Character Recognition (OCR) to extract the text contents from the attachments. If the intent of an email cannot be identified it is left for manual classification. Static rules and keywords gathered using feature engineering are used to classify emails (with high accuracy). Remaining emails are classified using an email classifiers based on machine learning.

The classifier model was built using a dataset of 260000 emails. Each email was preprocessed, using the strategies mentioned in the previous paragraph plus stop-words removal, punctuation removal, tokenization and lemmatization. During feature selection, ”To”, ”CC” and ”From” are removed and 180 custom features are created. Features from title, body and OCR texts are extracted using TF-IDF to represent 3-grams or using feature hashing. Then features are filtered using Chi-square (X^2) scoring.

Multi-class classification supervised algorithm were used as benchmarks (such as Random Forest and XGBoost). A hierarchical model was used, where some combination of feature extraction - feature selection - classifier was applied to some categories, deemed high accuracy categories, with the other categories, named low accuracy categories, grouped as one category. The next model predicts the low accuracy categories, using a given combination of feature extraction - feature selection - classifier that performs well to predict them.

Deep learning approaches were also employed. Long Short Term Memory unit (LSTM), Bidirectional Long Short Term Memory (Bi-LSTM) and Bidirectional Encoder Representations from Transformers (BERT) were tested, but despite being the state of the art, BERT was excluded because the accuracy obtained was not as good as expected and it has a high computer resources cost. LSTM and Bi-LSTM used GloVe embeddings.

In order to classify only high confident emails, thresholds for the categories were defined, based on the overall F-measure. High accuracy categories were set with zero threshold while low accuracy categories were set with a high threshold. If an email is classified with an accuracy lower than the threshold it is directed for manual classification.

Evaluation metrics such as precision, recall and F-measure were used. Despite the model with the best F-measure score was the LSTM with 77.3% F1 score, the hierarchical ML model using two models (one for high accuracy categories and other for low accuracy categories) combined with keyword static classification rules applied on ‘body+title+OCR’ with custom data engineered features with 76.5% F1 score was chosen due to being much less computationally expensive. With the deployment of this system, the human effort reduced from 100% to 20%. All the results are displayed on Table 3.1

Input Features	Classifier	Accuracy
Title	Random Forest	48.6
	XGBoost	48.6
	LSTM	52.1
	Bi-LSTM	52.3
Title + Body	Random Forest	62.9
	XGBoost	63.8
	LSTM	67.1
	Bi-LSTM	68.4
Title + Body + OCR	Random Forest	67.9
	XGBoost	69.1
	LSTM	73.2
	Bi-LSTM	74.6
Title + Body + OCR + Custom Engineered Features	Hierarchical ML	76.5
	('high accuracy categories' model)	83.2
	('low accuracy categories' model)	71.1
	Bi-LSTM	77.3

Table 3.1: [Shanmugalingam et al., 2019]'s Email Classification Models Performance

3.3 Similar Text Classification Problems

3.3.1 Traditional Classification Strategies

As early as 1999, [Yang and Liu, 1999] reported a comparative study on five categorization methods: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), NN, Linear Least-squares Fit (LLSF) and Naïve Bayes (NB). The corpus used was the longstanding benchmark *Reuters-21578*[1], namely the ApteMod version[2] which was obtained by eliminating unlabelled documents and selecting the categories which have at least one document in the training and test sets. 90 categories resulted in both a training set of 7769 documents and a test set of 3019 documents. The experiment settings were empirically selected. The feature selection technique used for each classifier was either Chi-square (X^2) or IG, according to which performed best. Each classifier feature set and settings are presented in the Table 3.2 with the results. By observing the results presented in Table 3.2, the conclusion is that SVM and KNN significantly outperform the other classifiers, while NB significantly under-performs all classifiers.

Since this study was reported, there have been a lot of research articles about improvements for traditional classifiers, hybrid approaches and new approaches.

The Neighbourhood Weighted KNN (NWKNN) [Tan, 2005] overcomes the problem of imbalanced datasets by assigning more weight for neighbors from small classes and less weight for neighbors from big classes.

One of the datasets used was a subset of the benchmark Reuters-21578, with 10324 documents under 55 categories. During the experiments, the documents were represented using the vector space

[1] https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection

[2] https://github.com/teropa/nlp/tree/master/resources/corpora/reuters

Method	config.	feat. size	miR	miP	miF1	maF1	error
SVM	linear model	10000	0.812	**0.914**	**0.860**	**0.525**	**0.00365**
KNN	k = 45	2415	0.834	0.880	0.857	0.524	0.00385
LLSF	singular value use for computation = 501	2415	**0.851**	0.849	0.850	0.501	0.00414
NN	54 units in the middle layer	1000	0.784	0.879	0.829	0.038	0.00447
NB		2000	0.769	0.825	0.796	0.389	0.00544

Table 3.2: Comparative Study on Reuters-21578 [Yang and Liu, 1999]

model (VSM) and the weight of each word was computed using TF-IDF. Features were selected using Information Gain and the dataset underwent three-fold cross validation.

The measures used were Recall, Precision and F1-measure. The results were evaluated for different values of exponent and selected features. It is worthy to note that with $exponent = 4$, NWKNN beats KNN by 5% under any number of features and achieves a F1-Measure of 68% and Precision and Recall of 69% and 70% respectively.

Hybrid approaches usually join the strengths of one or more individual approaches while trying to eliminate their drawbacks. While SVM can not identify document classes correctly when texts are positioned over the hyperplanes, KNN is limited when categorizing documents in overlapped category borders. Multi-class SVM-KNN (MSVM-KNN) [Yuan et al., 2008] overcomes these shortcomings and improves the performance of multi-class text classification by combining SVM and KNN, using SVM to identify class borders and KNN to categorize documents among those borders.

In this experiment, documents were represented by the VSM. Since TF-IDF (the most common approach for term weighting) does not consider a term discriminative power, [Yuan et al., 2008] proposed a new method to compute term weights, called TF-IDFH:

$$W_{tfidfh}(t,\overline{d}) = \frac{W_{tfidf}(t,\overline{d})}{H(t,\overline{d})} \tag{3.1}$$

where $W_{tfidf}(t,\overline{d})$ is the TF-IDF weight formula and $H(t,\overline{d})$ is an information entropy formula given by:

$$H(t,\overline{d}) = -\sum_{k=1}^{|C|} \frac{df_{ik}}{dn(t_i)} \log \frac{df_{ik}}{dn(t_i)} \tag{3.2}$$

To reduce dimensionality, a weight-based Best N feature selection was used, in which the top n terms with the highest information gain are selected to represent the document.

The authors implemented an ”Automatic Literature Categorization system (Automatic Literature Categorization (ALC))” [Yuan et al., 2008] to perform text categorization. The process of TC is composed of three stages, where in the first stage (Text Processing stage) ALC uses the vector space model to represent the documents, in the second stage (Training stage) ALC constructs MSVM-KNN classifier and in the third stage (Testing stage), the classifier classifies unlabeled documents.

Results were measured for the experiments on two datasets, 20-Newsgroups (19996 documents distributed among 20 classes) and an ACM dataset of 9 categories each with 25 documents. The measures used were Precision, Recall and F1-Measure. MSVM-KNN was compared with KNN and SVM. The F1-Measure of KNN, SVM and MSVM-KNN was 82%, 86%, and 90%, respectively, on 20-Newsgroup and 79%, 85%, and 89%, respectively, on the ACM dataset.

3.3.2 Hierarchical Classification Strategies

Hierarchical approaches, as mentioned earlier, may be grouped in two categories: local and global. As deep learning has been displaying superior performance over traditional classification approaches, deep learning models for text classification have been suggested. Hierarchical Deep Learning Text Classifier (HDLTex) [Kowsari et al., 2017] was presented to perform local (top-down) hierarchical classification. The HDLTex archictecture was built on top of three deep learning architectures, namely Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), where for each level of the label tree, one of the three DL architectures is chosen to compose the HDLTex architecture. In other words, for each level it trains each one of the three DL models. This means that for a n-level label tree, HDLTex model is then composed of n DL sub-models. The best HDLTex model is chosen empirically, meaning that from a n-level label tree, 3^n HDLTex models are built in order to choose the best one. The models use different feature extraction approaches. While CNN and RNN use text vector space models (using 100 dimensions), Deep Neural Networks (DNN) uses TF-IDF with counts for N-grams (which are sequences of N-words).

In contrast, the unified global hierarchical deep learning classifier [Sinha et al., 2018] overcomes this problem of the exponential grow in proportion to the number of tree levels. This model is composed of three parts: a Bi-LSTM encoder, an attention module and a MLP (Multi-Layer Perceptron). The Bi-LSTM extracts features of the documents in the form of word embeddings (300-dimensional word embeddings) followed by an attention module that supports the generation of dynamic document representations accross different levels of classification. At each level there is a two layered MLP, which predicts category at that level.

Both approaches for hierarchical TC were tested on the benchmark dataset *WOS-46985* so their results can be compared (Table 3.3). *Web of Science (WOS)* is a document collection of 46985 documents labelled under a hierarchical two-level taxonomy. The first level has 7 categories which contain {17, 16, 19, 9, 11, 53, 9} child categories (meaning that the second level has 134 categories in total). HDLTex submodels, two versions of SVM, a stacking SVM and Naive Bayes were used as baseline flat classifiers [Kowsari et al., 2017] as were also FastText-based classifier, Bi-LSTM with max/mean pooling and the Structured Self-attentive classifier [Sinha et al., 2018].

Method	**Accuracy**		
	I1	***I2***	**Overall**
HDLTex DNN [Kowsari et al., 2017]			66.95
HDLTex CNN [Kowsari et al., 2017]			70.46
HDLTex RNN [Kowsari et al., 2017]			72.12
Naïve Bayes			46.20
SVM			70.22
Stacking SVM			71.81
FastText-based			61.30
Bi-LSTM + MLP + Maxpool			**77.69**
Bi-LSTM + MLP + Maxpool			73.08
Structured Self Attention			77.40
HDLTex [Kowsari et al., 2017]	**90.45**	**84.66**	76.58
Global HDL [Sinha et al., 2018]	89.32	82.42	**77.46**

Table 3.3: Classification results of hierarchical approaches and traditional approaches on WOS-46985 [Sinha et al., 2018, Kowsari et al., 2017]

3.3.3 Other Classification Strategies

An Ontology is an explicit specification of a conceptualization and a formal way to define the semantics of knowledge and data. Normally, an ontology is developed to specify a particular domain (e.g., genetics). Such an ontology, often known as a domain ontology, formally specifies the concepts and relationships in that domain. The encoded formal semantics in ontologies is primarily used for effectively sharing and reusing of knowledge and data [Dou et al., 2015]. The formal definition of concepts and relationships in ontologies can be used to guide classification tasks, since it can specify consistency relationships of the classification task [Dou et al., 2015].

[Carvalho and Cohen, 2005] introduce the notion of email acts inspired by the Speech Act Theory [John, 1962, Searle and Searle, 1969]. These acts are described as ”noun-verb pairs that express typical intentions in email communication – for instance, to request for information, to commit to perform a task or to propose a meeting”. By matching these two concepts, ontology and email acts (in some works only described as ”intents”) some recent works in the subject of email categorization have proved better results regarding classification accuracy.

An email meeting intent classification ontology was built by [Cassier et al., 2019] from a set of 1150 annotated emails (with 458 containing at least a meeting intent and 692 without any meeting intent). Each of the 18 intent concepts contains lexical units and frame elements (which are Annotation Properties in the ontology) that allow detecting a specific intent in an email.

Emails are pre-processed and ontology lexical units and frame elements are projected in the email.

A score is computed by assigning a weight to units marked as mandatory in the ontology to select the three more likely intents. An annotation score is computed to each intent detected. Since the goal of [Cassier et al., 2019]'s work is to suggest answers to an email, intents are detected for each sentence of the email and an answer is generated according to each one.

Nevertheless, the effectiveness of intent classification in comparison to Machine Learning methods was tested and compared, using the same corpus of 1150 annotated emails. The ontology-based approach was compared with predictive models using Logistic Regression (LR), Decision Trees (DT), Random Forest (RF) and Naïve Bayes (NB) algorithms with a Bag of Words (BoW) approach scored with TF-IDF. The ontology-based approach scored 0.71, 0.67 and 0.69 on Precision, Recall, F1-Measure, respectively, while the best of the ML methods score 0.25, 0.23 and 0.24, respectively.

3.4 Summary

From the few works regarding Email Classification to the more general TC (Text Classification), the works described in this chapter show strategies applied to classify small datasets of personal emails, big datasets of corporate emails, incident tickets (which have a structure similar to emails), (benchmark) datasets of news articles and (benchmark) datasets of scientific articles, which have a structure composed of introduction, development and conclusion and one ore more intents.

All these studies, despite their differences, converge on some points. Most of them use TF-IDF for feature extraction and Chi-square (X^2) for feature selection. Studies who mention preprocessing mention stop-word removal, tokenization and stemming or lemmatization.

Either as the target classifiers or as benchmarks/baselines, ML algorithms are always part of the classifiers presented in these studies (with exception of the ones presented in section 3.3.3). Deep Learning algorithms are used with success in more recent studies, but despite their (usually) superior performance, ML algorithms present competitive alternatives, due to close performance, to the simplicity of their implementation and to the low computational cost. Table 3.4 shows a comparison between the studies presented in sections 3.2 and 3.3.

However, there is a big number of limitations in these studies when compared to our work. The only datasets that regard emails, with the exception of the one reported by [Shanmugalingam et al., 2019], are small (contain around 1000 emails) and the emails are personal and have been classified to a small set of classes. The benchmarks datasets (*Reuters* and *WoS*) are well studied and well refined for classification purposes. Furthermore, the dataset reported by [Shanmugalingam et al., 2019] and *WoS* are the only ones that have a hierarchy, namely a ”balanced” hierarchy with two and three levels (respectively), where all the target classes are leaf-nodes at the same level and with categories that have an objective semantic: each category represents a subject/domain.

Author	Data Type	Preprocessing	Feat. Extraction	Feat. Selection	Classifiers
[Cohen, 1996]	Personal Emails	Key-word selection	TF-IDF	Select only the first 100 words	Rule-based, Custom classifier
[Yang and Liu, 1999]	News articles (Reuters-21578)	(Not mentioned)	TF-IDF	Chi^2, IG	SVM, LLSF, KNN, NN, NB
[Clark et al., 2003]	Personal Emails	Tokenization, Remove words based on length and frequency	BOW, TF-IDF, TF, Term Presence	IG, Variance	MLP
[Tan, 2005]	News articles (Reuters-21578)	(Not mentioned)	TF-IDF	IG	Neighbor Weighted KNN
[Xia et al., 2007]	Emails (Enron dataset)	Stop-words Removed	TF-IDF	SFS (based on TruncatedSVD)	SVMs, DT
[Yuan et al., 2008]	News articles (Reuters-21578)	(Not mentioned)	TF-IDF, TF-IDF variant	(Not mentioned)	SVM, KNN and MSVM-KNN (SVM and KNN variant)
[Yu and hua Zhu, 2009]	Personal emails	(Not mentioned)	(Not mentioned)	SFS (based on TruncatedSVD)	Custom MLP
[Kowsari et al., 2017]	Scientific articles (WoS)	Word Embeddings	TF-IDF	(Not mentioned)	HDLTex (DNN + CNN + RNN)
[Sinha et al., 2018]	Scientific articles (WoS)	Word Embeddings	(Not mentioned)	(Not mentioned)	Global Hierarchical DL classifier (Bi-LSTM + MLP)
[Silva et al., 2018]	IT incident tickets	Tokenization, Stemming, Stop-words rem.	TF-IDF, TF, IDF, NER	(Not mentioned)	SVM, KNN
[Cassier et al., 2019]	Personal Emails	(Not mentioned)	TF-IDF	(Not mentioned)	LR, DT, NB, Random Forest, Ontology-based rules
[Shanmugalingam et al., 2019]	Corporate emails	Stop-words removed, key-body content selection, OCR	TF-IDF	Chi^2	XGBoost, LSTM, Bi-LSTM, BERT, Random Forest, Custom rules, Custom Hierarchical Model

Table 3.4: Related Works Summary

4

Data

If you torture data long enough, it will confess to anything.

– Ronald H. Coase, renowned British economist

Contents

4.1 Domain Understanding

From the business (contact center) perspective, the objective of this work is to increase email management and processing efficiency by having an automatic email classification system that aids in the classification of received emails.

In the context of the Contact Center, emails flow as represented in Fig. 1.1. Currently, emails are processed by a group of seven agents. Emails are classified over a set of predefined classes organized in a hierarchical structure. The objective of a classifier (human or artificial) is to classify each received email with a leaf-node category from the hierarchy tree. Each leaf-node category corresponds to a set of predefined answers (generally 1 to 4 answers). A non-leaf category represents all the answers of its child categories. A predefined answer may be present in more than one leaf-node category. Generally, top-level nodes share almost no answer while sibling leaf-node categories may share most of their answers.

The contact center system allows an email to be classified with an inter-level category or even not to be classified at all and to be replied in this condition, since the goal of the contact center email management system is to answer emails, not to classify emails. The classification step serves two purposes:

1. To organize emails over a taxonomy that has semantic meaning;
2. To reduce the effort needed to find the right answer to reply the email.

Classification and Reply are distinct steps, meaning that an email does not need to be replied right after it has been classified. A classified email allows it to be picked by a contact center operator that is an expert in the category domain it has been classified with and answer it.

In conclusion, the main goal of the classification step is to increase the efficiency by which an email is answered, by reducing the possible answers it may have and by allowing it to be easily picked by a domain expert. In the category hierarchy tree, the top categories have a bigger importance than the lower categories, since they have more impact in distinguishing the email domain and lower level categories end up having more of a ”stage of email subject” or description role (see Figure 4.1).

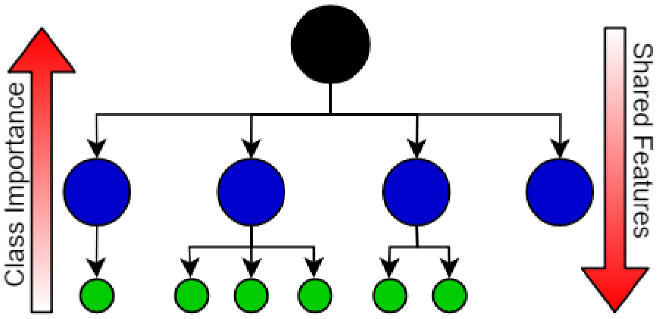


Figure 4.1: Hierarchical tree level importance / class distinguishing power

4.2 Data Exploration and Cleaning (Data Understanding)

4.2.1 Data Acquisition and Exploration

Since data handled in the contact center is of sensitive nature, collecting data for the development of this work was all but straightforward. To collect data, a request had to be sent, data underwent an anonymization process defined by the contact center and then it was delivered to us.

All data collected and delivered was in raw format, organized in JavaScript Object Notation (JSON) files, each one representing a chain of emails exchanged between a Contact Center agent and a citizen, called ”ticket” in the context of the Contact Center.

The process of interpreting the raw data in the JSON was hindered by two aspects:

1. **Documentation:** There was not a document explaining the organization of the data in the JSON file. This problem disabled the possibility of an automatic approach to analyse the data and required a manual and careful analysis of the data to distinguish the meaningful data from the useless data.
2. **Anonymization:** The anonymization strategies that where applied where mainly based on suppression of sensitive data which rendered data less readable and understandable.

Each ticket's JSON was inferred to be organized as described in appendix B.1, where the “episodes” compile a list of ticket actions. Each ticket episode refers to an action discriminated by “type”. The subject of each email was stored in JSON property *subject* (it had the same value for all emails within a chain of emails), email body contents were inferred to be in the “text_part” or “html_part” of episodes with ‘type: *INBOUND*’ or ‘status: *NEW*’ and the category assigned to each email was inferred to be in the field “episode_subject” of the first following episode with ‘status: *FW_WAIT*’ and ‘type: *OUTBOUND*’, or just ‘status: *CLOSED*’. “To” and “From” features of each email where anonymized.

Despite the challenges, a dataset was built in a Comma-Separated Values (CSV) format with the main attributes “Text”, “Subject” and “Class”, and metadata attributes “Reply Number” and “Ticket”. The index of each entry was formed by the concatenation of the “Ticket” value and “Reply Number”. The ticket identifies a chain of emails. The reply number identifies the position of the email in the ordered chain of emails (“Ticket”). “Text” is the email content and was extracted from the “text_part” and “html_part” JSON fields. “Subject” is the subject of the email and was extracted from the “subject” JSON field.

The dataset built from extracting data from the JSON files retrieved at the data acquisition process is called raw dataset and named with the suffix ‘*_raw*’. During the data understanding phase, the raw dataset is pre-processed, using data cleaning and normalization strategies, in order to remove noise, standardize values and increase the percentage of useful, meaningful data. The dataset obtained at the end of this phase will be subject to data/feature representation strategies in the data preparation phase, that will prepare the dataset for the modeling phase.

During the execution of this project and due to the intricacies of acquiring new data, there were three data collection actions. These actions happened, in the scope of the CRISP-DM methodology, after a cycle was executed and in the evaluation phase was decided that more data (or better data) was needed in order to achieve better results.

The first collection of data had 11898 emails distributed over 102 categories. The second collection of data added 26368 new emails and 25 categories not recorded before, resulting in 38266 documents and 127 categories. The third collection of data added 11298 new emails and 11 categories not recorded before, resulting in 49564 documents and 138 categories.

In order to simplify the management of the datasets that resulted from each iteration, they were named **Av1**, **Av2** and **Av3** (first collection/iteration, second collection/iteration and third collection/iteration, respectively). The datasets at the point before the data cleaning and normalization process are named **Av1_raw**, **Av2_raw** and **Av3_raw**. Av2_raw and Av2 are the same because at that iteration, the aim of this process was only to clean and not to normalize data and that cleaning process happened before data acquisition. Dataset terminology is summarized in Table 4.1.

Iteration	Dataset Name	Instances count	Class count	Vocabulary Size	Noise Cleaning	Data Normalization
Iteration 1	Av1_raw	11898	102	27578		
Iteration 1	Av1	11898	102	27491	X	
Iteration 2	Av2/Av2_raw	38266	127	73529	X	
Iteration 3	Av3_raw	49564	138	103694	X	
Iteration 3	Av3	49564	138	65408	X	X

Table 4.1: Datasets Info

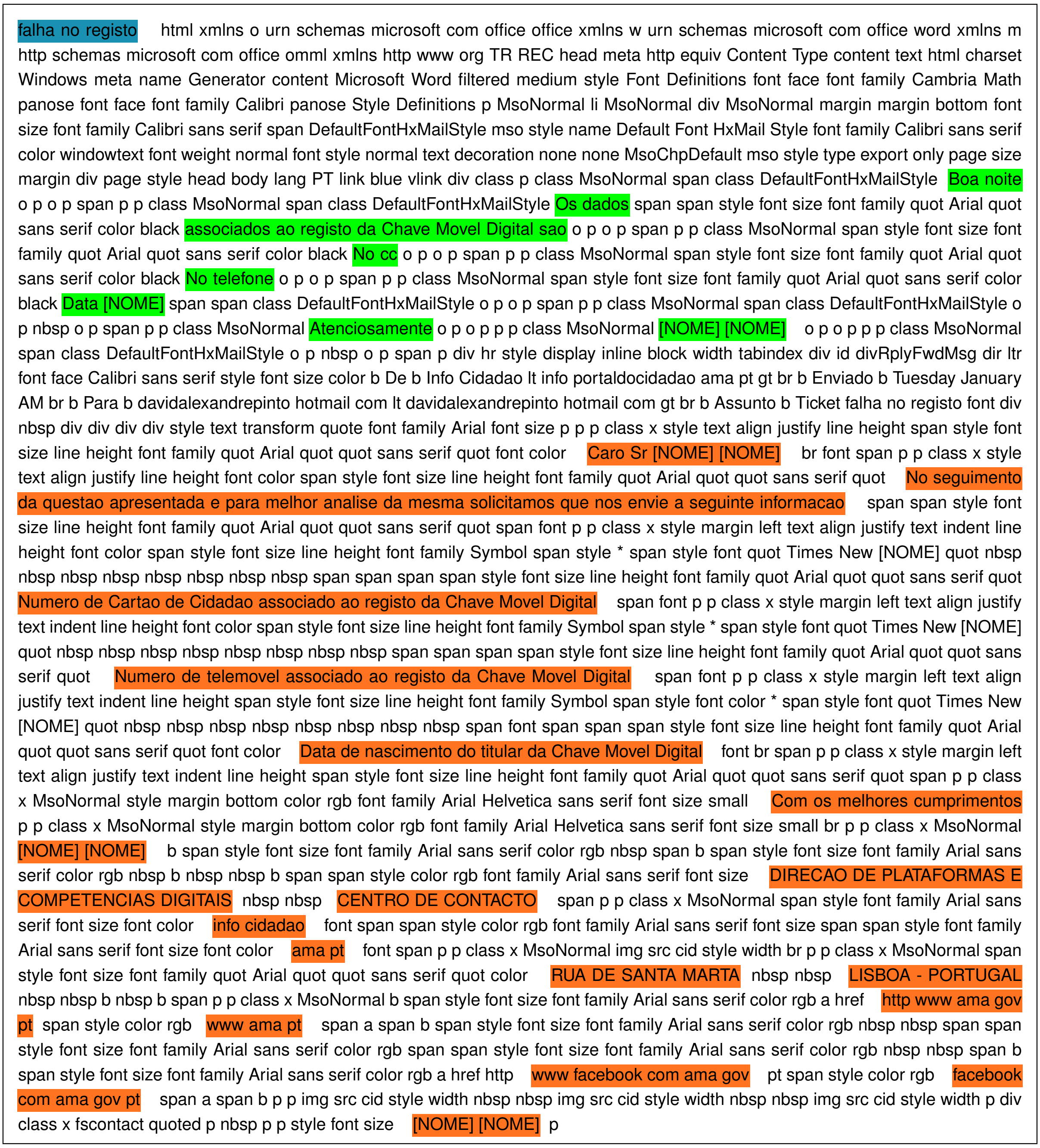
4.2.2 Data Noise

As mentioned in section 2.1 data noise is one of the biggest obstacles to text classification. The example in Figure 4.2 shows that at first contact with the data we had more noise than data.

The data noise portrayed in this sample and through all data may be categorized by three sources of noise:

The business process (systematic noise): Almost every email is plagued with a huge proportion of noise, mainly in the format of markup language which may be a product of the data storage or data extraction from the platform where they are managed and processed. Also, the fact that the email carries the email it is answering to, is a source of noise, since this is unwanted data.

The anonymization process (systematic noise): The anonymization strategies applied (which were decided by the contact center) were mainly based on suppression of sensitive data. The initial rules applied (at first iteration) were to remove all punctuation, remove all numbers and replace each name with a placeholder '[NOME]'. As Table 4.2 shows, this negatively impacted both noise and meaningful



falha no registo html xmlns o urn schemas microsoft com office office xmlns w urn schemas microsoft com office word xmlns m http schemas microsoft com office omml xmlns http www org TR REC head meta http equiv Content Type content text html charset Windows meta name Generator content Microsoft Word filtered medium style Font Definitions font face font family Cambria Math panose font face font family Calibri panose Style Definitions p MsoNormal li MsoNormal div MsoNormal margin margin bottom font size font family Calibri sans serif span DefaultFontHxMailStyle mso style name Default Font HxMail Style font family Calibri sans serif color windowtext font weight normal font style normal text decoration none none MsoChpDefault mso style type export only page size margin div page style head body lang PT link blue vlink div class p class MsoNormal span class DefaultFontHxMailStyle Boa noite o p o p span p p class MsoNormal span class DefaultFontHxMailStyle Os dados span span style font size font family quot Arial quot sans serif color black associados ao registo da Chave Movel Digital sao o p o p span p p class MsoNormal span style font size font family quot Arial quot sans serif color black No cc o p o p span p p class MsoNormal span style font size font family quot Arial quot sans serif color black No telefone o p o p span p p class MsoNormal span style font size font family quot Arial quot sans serif color black Data [NOME] span span class DefaultFontHxMailStyle o p o p span p p class MsoNormal span class DefaultFontHxMailStyle o p nbsp o p span p p class MsoNormal Atenciosamente o p o p p p class MsoNormal [NOME] [NOME] o p o p p p class MsoNormal span class DefaultFontHxMailStyle o p nbsp o p span p div hr style display inline block width tabindex div id divRplyFwdMsg dir ltr font face Calibri sans serif style font size color b De b Info Cidadao lt info portaldocidadao ama pt gt br b Enviado b Tuesday January AM br b Para b davidalexandrepinto hotmail com lt davidalexandrepinto hotmail com gt br b Assunto b Ticket falha no registo font div nbsp div div div div style text transform quote font family Arial font size p p p class x style text align justify line height span style font size line height font family quot Arial quot quot sans serif quot font color Caro Sr [NOME] [NOME] br font span p p class x style text align justify line height font color span style font size line height font family quot Arial quot quot sans serif quot No seguimento da questao apresentada e para melhor analise da mesma solicitamos que nos envie a seguinte informacao span span style font size line height font family quot Arial quot quot sans serif quot span font p p class x style margin left text align justify text indent line height font color span style font size line height font family Symbol span style * span style font quot Times New [NOME] quot nbsp nbsp nbsp nbsp nbsp nbsp nbsp nbsp span span span span style font size line height font family quot Arial quot quot sans serif quot Numero de Cartao de Cidadao associado ao registo da Chave Movel Digital span font p p class x style margin left text align justify text indent line height font color span style font size line height font family Symbol span style * span style font quot Times New [NOME] quot nbsp nbsp nbsp nbsp nbsp nbsp nbsp nbsp span span span span style font size line height font family quot Arial quot quot sans serif quot Numero de telemovel associado ao registo da Chave Movel Digital span font p p class x style margin left text align justify text indent line height span style font size line height font family Symbol span style font color * span style font quot Times New [NOME] quot nbsp nbsp nbsp nbsp nbsp nbsp nbsp nbsp span font span span span style font size line height font family quot Arial quot quot sans serif quot font color Data de nascimento do titular da Chave Movel Digital font br span p p class x style margin left text align justify text indent line height span style font size line height font family quot Arial quot quot sans serif quot span p p class x MsoNormal style margin bottom color rgb font family Arial Helvetica sans serif font size small Com os melhores cumprimentos p p class x MsoNormal style margin bottom color rgb font family Arial Helvetica sans serif font size small br p p class x MsoNormal [NOME] [NOME] b span style font size font family Arial sans serif color rgb nbsp span b span style font size font family Arial sans serif color rgb nbsp b nbsp nbsp b span span style color rgb font family Arial sans serif font size DIRECAO DE PLATAFORMAS E COMPETENCIAS DIGITAIS nbsp nbsp CENTRO DE CONTACTO span p p class x MsoNormal span style font family Arial sans serif font size font color info cidadao font span span style color rgb font family Arial sans serif font size span span style font family Arial sans serif font size font color ama pt font span p p class x MsoNormal img src cid style width br p p class x MsoNormal span style font size font family quot Arial quot quot sans serif quot color RUA DE SANTA MARTA nbsp nbsp LISBOA - PORTUGAL nbsp nbsp b nbsp b span p p class x MsoNormal b span style font size font family Arial sans serif color rgb a href http www ama gov pt span style color rgb www ama pt span a span b span style font size font family Arial sans serif color rgb nbsp nbsp span span style font size font family Arial sans serif color rgb span span style font size font family Arial sans serif color rgb nbsp nbsp span b span style font size font family Arial sans serif color rgb a href http www facebook com ama gov pt span style color rgb facebook com ama gov pt span a span b p p img src cid style width nbsp nbsp img src cid style width nbsp nbsp img src cid style width p div class x fscontact quoted p nbsp p p style font size [NOME] [NOME] p

Figure 4.2: Av1_raw (First dataset of first data collection) - Email Sample (blue: subject; green: body content; orange: email that is being replied)

data, making meaningfull data harder to identify (even for a human) and data and noise harder to distinguish.

Human writing (natural noise): Each chain of emails ('Ticket') is written by a different person, which means that the vocabulary is far from being standardized, that emails have been written by both persons with education and without education, old and new, and will show many misspellings,

Before Anonymization	After Anonymization
🟥 style font family quot Times New Roman	🟥 style font family quot Times New [NOME]
🟥 Morada Av Estados Unidos da América	🟥 Morada Av Estados Unidos da [NOME]
🟩 Caro Sr Luís Neto	🟩 Caro Sr [NOME] [NOME]
🟥 Caro Sr Fernando Haley	🟥 Caro Sr [NOME] Haley
🟥 To fhaley sapo pt	🟥 To fhaley sapo pt
🟩 Numero de Cartao de Cidadao 12345678	🟩 Numero de Cartao de Cidadao
🟥 Sent Wed Set	🟥 Sent Wed [NOME]

Table 4.2: Examples of anonymization performed on the data (green - correct; red - unwanted/incorrect/uncompleted

wrong words, ill formed sentences, synonyms, different sentence structures for the same idea, etc.

The problem of having unrequested information (the email that was being replied to) in the email was largely ignored during the development of this work mainly because the harshness and importance of the two other problems (anonymization errors and noise) overshadowed this problem and to find a strategy to distinguish two emails in unstructured data such as this would be very laborious.

During the first iteration, noisy data was removed by manually identifying frequent blocks of markup language and removing them. This strategy, although very tedious and time-consuming, was chosen for three reasons:

1. This was the first contact with this undescribed and undocumented data, so a manual approach was required in order to know the data.
2. Meaningful data was only a fragment of the email content and automatic approaches could indirectly and undesirably reduce it. Noise expressions where carefully identified in order not to include any possible meaningful information.
3. Anonymization introduced meaningful terms (such as '[NOME]' in 'style font quot Times New [NOME]') within markup language expressions and any (to the best of our knowledge) automatic markup language noise removal strategy would at the very best case, leave these terms in the text, further reducing text readability

From the second iteration on, (at our request) the anonymization process changed: data markup language was cleaned using *beautifulsoup*[1] python library, before the anonymization approaches, in order to remove markup language noise. This strategy was much more effective and efficient than the manual strategy, and although it removed almost all of the markup language noise, it could not remove all of it. The anonymization strategies also changed, at our request, at each data collection iteration, in order to to be less restrictive and increase data readability and meaningfulness. Table 4.3 summarizes the results of the anonymization and noise removal strategies applied at each iteration.

[1] https://pypi.org/project/beautifulsoup4/

Av1	Av2/Av2_raw	Av3_raw
Anonymization: 1. Remove punctuation 2. Replace names with placeholder [NOME] 3. Remove all numbers Noise cleaning: Manual identification and removal of noise expressions	Anonymization: 1. Replace names with placeholder [NOME] 2. Remove all numbers Noise cleaning: *BeautifulSoup* python library	Anonymization: 1. Replace names with placeholder [NOME] 2. Replace all numbers with a placeholder [NUMEROX], where X is the number of digits in the number. Noise cleaning: *BeautifulSoup* python library
falha no registo DefaultFontHxMailStyle name Default Font HxMail Style DefaultFontHxMailStyle Boa noite DefaultFontHxMailStyle Os dados associados ao registo da Chave Movel Digital sao No cc No telefone Data [NOME] DefaultFontHxMailStyle DefaultFontHxMailStyle Atenciosamente DefaultFontHxMailStyle De Info Cidadao info portaldocidadao Enviado Tuesday January AM Para davidalexandrepinto hotmail com davidalexandrepinto hotmail com Assunto Ticket falha no registo Caro Sr No seguimento da questao apresentada e para melhor analise da mesma solicitamos que nos envie a seguinte informacao * Numero de Cartao de Cidadao associado ao registo da Chave Movel Digital * Numero de telemovel associado ao registo da Chave Movel Digital * Data de nascimento do titular da Chave Movel Digital Com os melhores cumprimentos DIRECAO DE PLATAFORMAS E COMPETENCIAS DIGITAIS CENTRO DE CONTACTO info cidadao RUA DE SANTA MARTA LISBOA - PORTUGAL a a p	falha no registo boa [NOME] , os dados associados a o registo da chave movel digital sao : no cc : no telefone : data [NOME] : atenciosamente , [NOME] [NOME] de : info cidadao enviado : tuesday , january am para : assunto : [ticket falha no registo caro sr . [NOME] [NOME] , no seguimento da questao apresentada , e para melhor analise da mesma , solicitamos que nos envie a seguinte informacao : * numero de cartao de cidadao associado a o registo da chave movel digital ; * numero de telemovel associado a o registo da chave movel digital ; * data de [NOME] do titular da chave movel digital . com os melhores cumprimentos , [NOME] [NOME] — direcao de plataformas e competencias digitais — centro de contacto [NOME] de santa [NOME] , — [NOME] - [NOME] — + www.ama.pt/ — facebook.com/ama.gov.pt - [NOME] [NOME] :	falha no registo boa [NOME] , os dados associados a o registo da chave movel digital sao : no cc : NUMERO8 NUMERO1 zz1 ; no telefone : 916280131 ; data [NOME] : 16-02-1960 . atenciosamente , [NOME] [NOME] de : info cidadao enviado : tuesday , january 14 , NUMERO4 9:52:15 am para : assunto : [ticket gx4qn6] falha no registo caro sr . [NOME] [NOME] , no seguimento da questao apresentada , e para melhor analise da mesma , solicitamos que nos envie a seguinte informacao : * numero de cartao de cidadao associado a o registo da chave movel digital ; * numero de telemovel associado a o registo da chave movel digital ; * data de [NOME] do titular da chave movel digital . com os melhores cumprimentos , [NOME] [NOME] — direcao de plataformas e competencias digitais — centro de contacto [NOME] de santa [NOME] , NUMERO2 — 1150-294 [NOME] - [NOME] — + NUMERO3 NUMERO3 NUMERO3 NUMERO3 www.ama.pt/ — facebook.com/ama.gov.pt 14-01-2020 01:19:10 - [NOME] [NOME] :

Table 4.3: Email sample at iterations 1, 2 and 3 after anonymization and noise removal (blue: subject; green: body content; orange: email that is being replied)

One of the advantages of the manual noise removal strategy was that by close inspection of data, special cases of normalization such as 'Boa [NOME]' to 'Boa Noite' (observed at the Table 4.3) could be identified and corrected.

The goal of the anonymization strategy at third iteration was to augment the meaningful information of the data by applying a softer strategy instead of strictly removing all the number's digits. As the third example of Table 4.3 shows, this new strategy was not successfully applied and brought the following issues:

1. Anonymization false positives: Some sensitive data was left known.
2. Lack of normalization: Only the numbers that where surrounded by space characters where transformed.
3. It increased not only the amount of meaningful data, but also the amount of meaningless data

In order to further remove noise and normalize data, an extensive set of operations, mainly based on regular expressions where applied at the third iteration first dataset. Types of noise and normalization issues identified, the respective solution and examples are described in Table 4.4 The application of these operations resulted in the final dataset of the third iteration, represented in Table 4.5.

4.2.3 Data Description

The dataset extracted from the raw data has the following attributes:

Class is a string representing the category of the email. The string may have slashes ”/” which separate the hierarchical levels. Example: Class value ”Info Cidadão/IRN/Informações Gerais” means that level 1 class is ”Info Cidadão”, level 2 class is ”IRN”, and level 3 ”Informações Gerais”.

Text is a string representing the email body content. It is unstructured data and the main source of features.

Subject is a string representing the email subject.

Ticket is a string identifying a chain of emails. For a given ticket value the value of the subject is constant.

Reply Nr is an integer representing the position of the email in the ordered chain of emails.

The *spaCy* [Honnibal and Montani, 2017] python library was used during the data preparation phase with the purpose of performing lemmatization and to extract other useful features such as verbs and nouns. Loaded with a Portuguese language large model ‘pt_core_news_lg’, a part-of-speech analysis was performed to get a description of data and to visualize the evolution of data between iterations and different stages of data cleaning and normalization.

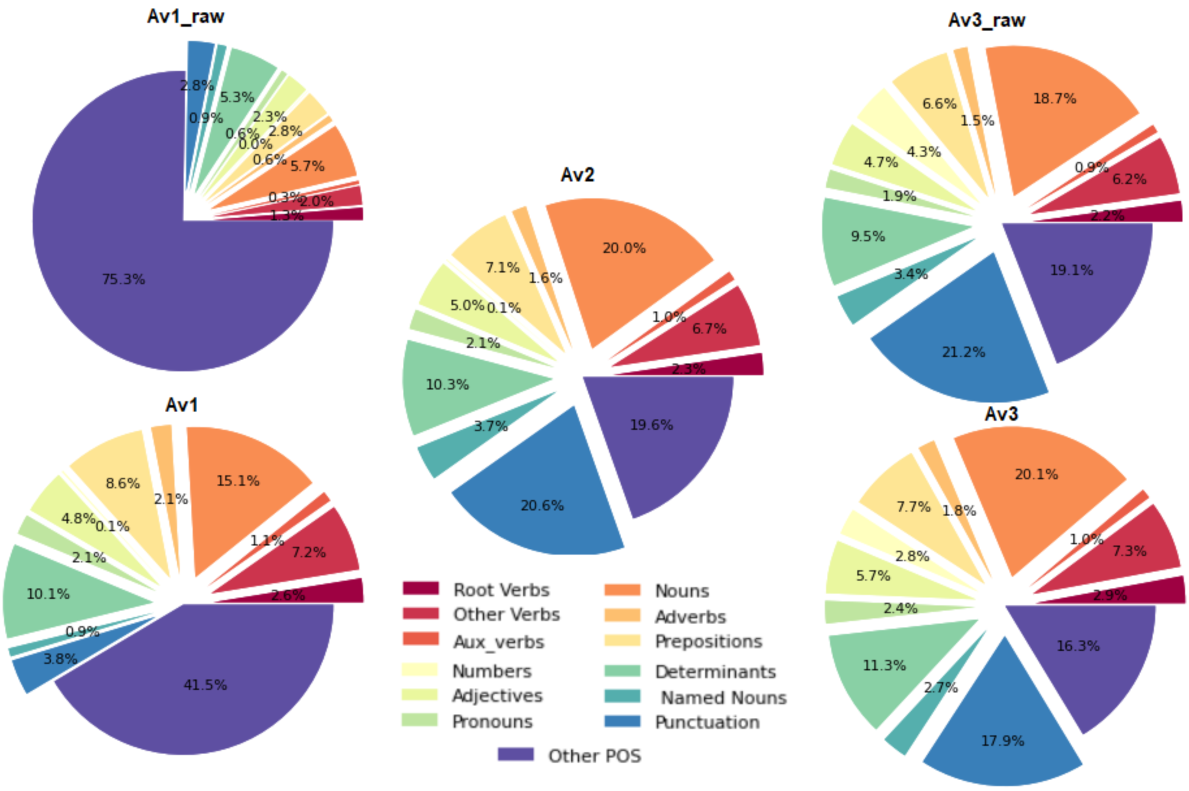


Figure 4.3: Evolution of Average Identified POS Tags per Email (using *spaCy ´pt_core_news_lg’* model)

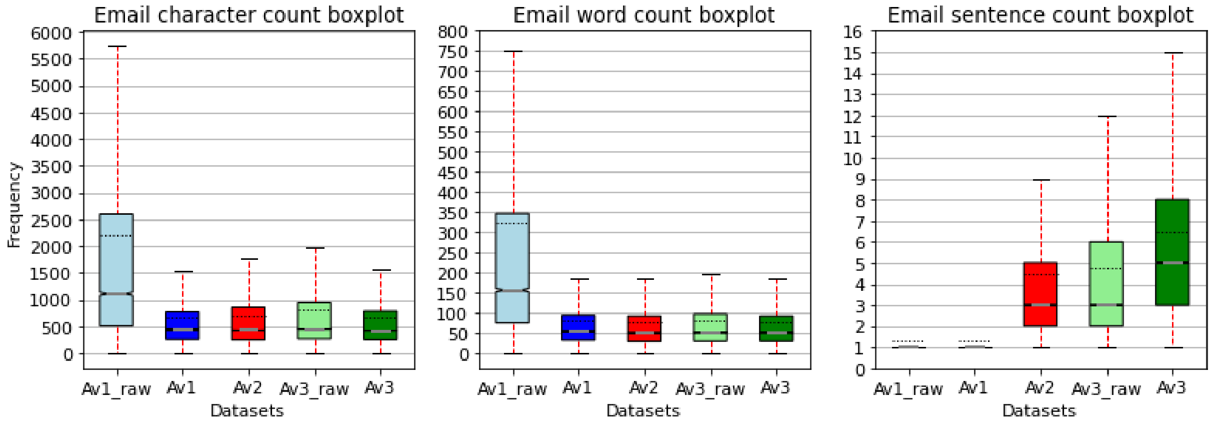


Figure 4.4: Evolution of the Frequency of Characters, Words and Sentences in Emails by Dataset

Figure 4.3 shows the results, where a significant improvement of the meaning of data can be perceived. Figure 4.4 shows three box plot graphs where the effects of data cleaning are also visible. The first two graphs show most noticeable the difference in the amount of data between the dataset without any cleaning (Av1_raw) and the others. The mean measure in each boxplot, represented by the black dotted line, shows a value approximately four times higher in Av1_raw than in the other datasets, suggesting that 75% of the original data was systematic noise. There is a slight increase in Av3_raw character and word frequency which is explained by the softening of the anonymization rules at the third iteration, which did not remove numbers. In the third graph, we notice that we could not identify sentences at iteration one, because all the punctuation was removed. It also shows that the noise cleaning and data normalization strategies applied (presented at Section 4.2.4) allowed a better representation of data (most noticeable, the solution applied to the problem of "Words/numbers concatenated by punctuation" (see Table 4.4).

4.2.4 Data Cleaning

The first data quality problem identified was noise. Probably due to the framework where emails were managed or because of the extraction process, the email content was mixed with markup language. Furthermore, since each chain of emails is written by a different person the vocabularies are not standardized. There are synonyms, misspellings, random punctuation and different sentence structures to express the same idea/meaning.

Due to this scenario, strategies to clean and normalize data were defined, comprehending the following steps:

1. Explore data;
2. Define data cleaning and normalization objectives;

Noise/ Normalization Problem	**Solution**	**Example (red = original, green = normalized)**
Citizen card number suffix	1. Identified citizen card number suffix patterns; 2. Replace every match with the placeholder '[CCSUF]', using regular expressions.	'NUMERO8 NUMERO1 zy7tlm' 'NUMERO8 [CCSUF] tlm'
Misspelings	1. Parse all data using spaCy's model "pt_core_news_lg" to identify OOVs. 2. Identify the most frequent misspellings from the OOVs and replace them in all data instances with the correct form.	'pasaporte', 'passporte', 'pasaport', 'passaport', 'passaportee', 'passaporte'
Sequences of placeholder '[NOME]'	Using regular expressions, reduce all sequences of '[NOME]' to just one '[NOME]'.	'cumprimentos, [NOME] [NOME] [NOME]', 'cumprimentos, [NOME]'
Sequences of placeholder 'NUMEROx'	Using regular expressions, reduce all sequences of 'NUMEROx' to just one 'NUMEROy', where for a sequence 'NUMEROx_1 NUMEROx_2 NUMEROx_3 NUMEROx_3' we have $y = x_1 + x_2 + x_3 + x_4$	'tlm: NUMERO2 NUMERO4 NUMERO3', 'tlm: NUMERO9'
Time and Date	Identify the patterns of time and date and replace them by '[HORA]' and '[DATA]', respectively, using regular expressions.	'02/01/2020 16:04:56 (utc)', 'NUMERO1 dec NUMERO4 23:24:09 +0000' '[DATA] [HORA]'
Sequences of punctuation	Using regular expressions, reduce all sequences of punctuation to one occurrence.	'justo???' 'justo?'
IP addresses and websites	Using regular expressions, remove all IP addresses and websites.	'from [194.38.147.41]' 'from [ ]'
Special characters	Using regular expressions, remove all characters that are not numbers, letters, normal punctuation ('.', '!', '?', ',', '-', '+') or part of the placeholders ('[', ']').	'$', '#', '[ ]'
Words concatenated with numbers	Using regular expressions, all words that had a sequence of numbers followed by a sequence of letters were split by a space character and the numbers were replaced by the appropriate placeholder.	'tlm915321456' 'tlm NUMERO9'
Words/numbers concatenated by punctuation	Using regular expressions, all words or numbers concatenated by punctuation were split with a space character between the punctuation and each sub-word. In case one sub-word is a number, it was replaced by the appropriate placeholder.	'cartão.amanhã', 'cc.18326979' 'cartão . amanhã', 'cc . NUMERO9'
Concatenated words	1. Parse all data using spaCy's model "pt_core_news_lg" to identify OOVs. 2. When a OOV is identified, all possible divisions of that word are tested in order to find a division where both words were recognized by the vocabulary. 3. If successful, the original word and the resulting pair are stored in a python dictionary, with the original word as a key and the resulting pair as a value. 4. After a manual analysis of the dictionary to verify the results, all keys identified were replaced in the data instances by the resulting pair separated by a space.	'cumprimentosluis', 'titularnif' 'cumprimentos [NOME]', 'titular nif'
Hash words and html content	1. Most frequent html related and hash words are manually identified from the OOVs added to a stop-word list. 2. Remove everything between two brackets ('{', '}') 3. Remove all words with a length superior to 25 characters, since there are only a few words in Portuguese longer than 25 characters (this was done after dealing with the concatenations of words) 4. Remove words with given patterns: two or more sequences of numbers and letters or words with special characters ('$', '', '+', '%', '')	' table.sample { border-width ... , 240) ;}' 'table.sample'

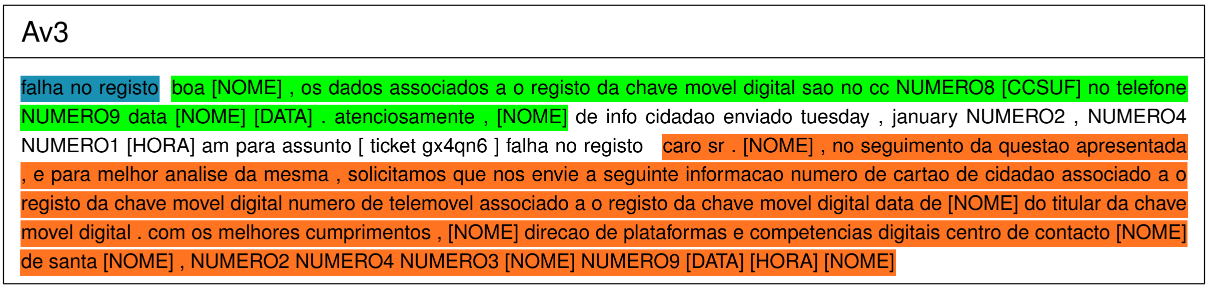
Table 4.4: Noise Removal and Normalization Approaches

3. Implement a strategy to clean/normalize the data;
4. Analyze the results.

Strategies to clean/normalize data are usually centered around the application of regular expressions that parse all data and when it matches a given pattern a transformation is applied. Regular expressions are simple to write. In order to ensure a successful application of regular expressions, their results were analysed carefully almost as a data exploration task. The analysis of the results led to the discovery of new normalization opportunities.

To find more noise examples we used *spaCy*[2] python library. The large model 'pt_core_news_lg'[3], contatining a vocabulary of 642875 unique words, trained on UD Portuguese Bosque[4], WikiNER[5], OSCAR (Common Crawl)[6] and Wikipedia[7], was used to parse all emails and list all the Out Of Vocabularys (OOVs) (Out Of Vocabulary words) and their frequencies. The output of this strategy was a python dictionary with all the words that were not recognized by the spaCy model vocabulary as keys and their frequency as value, ordered by value.

From this python dictionary, we were able to study the most frequent OOVs and design several strategies to deal with them, which are described in Table 4.4. Figure 4.5 shows the result of the normalization strategies applied on the Av3_raw example portrayed in Table 4.3.



Av3

falha no registo boa [NOME] , os dados associados a o registo da chave movel digital sao no cc NUMERO8 [CCSUF] no telefone NUMERO9 data [NOME] [DATA] . atenciosamente , [NOME] de info cidadao enviado tuesday , january NUMERO2 , NUMERO4 NUMERO1 [HORA] am para assunto [ticket gx4qn6] falha no registo caro sr . [NOME] , no seguimento da questao apresentada , e para melhor analise da mesma , solicitamos que nos envie a seguinte informacao numero de cartao de cidadao associado a o registo da chave movel digital numero de telemovel associado a o registo da chave movel digital data de [NOME] do titular da chave movel digital . com os melhores cumprimentos , [NOME] direcao de plataformas e competencias digitais centro de contacto [NOME] de santa [NOME] , NUMERO2 NUMERO4 NUMERO3 [NOME] NUMERO9 [DATA] [HORA] [NOME]

Figure 4.5: Av3 (final dataset of third data collection/iteration) - Email Sample (blue: subject; green: body content; orange: email that is being replied)

4.2.5 Class Taxonomy

Prior to the data exploration tasks, it was already known from the business understanding phase that the email classes were organized in a hierarchical structure. The structure is represented in Figure 4.6.

From Figure 4.6 we can make the following observations:

[2] https://spacy.io/
[3] https://spacy.io/models/ptpt_core_news_lg
[4] https://github.com/UniversalDependencies/UD_Portuguese-Bosque
[5] https://figshare.com/articles/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500
[6] https://oscar-corpus.com/
[7] https://dumps.wikimedia.org/

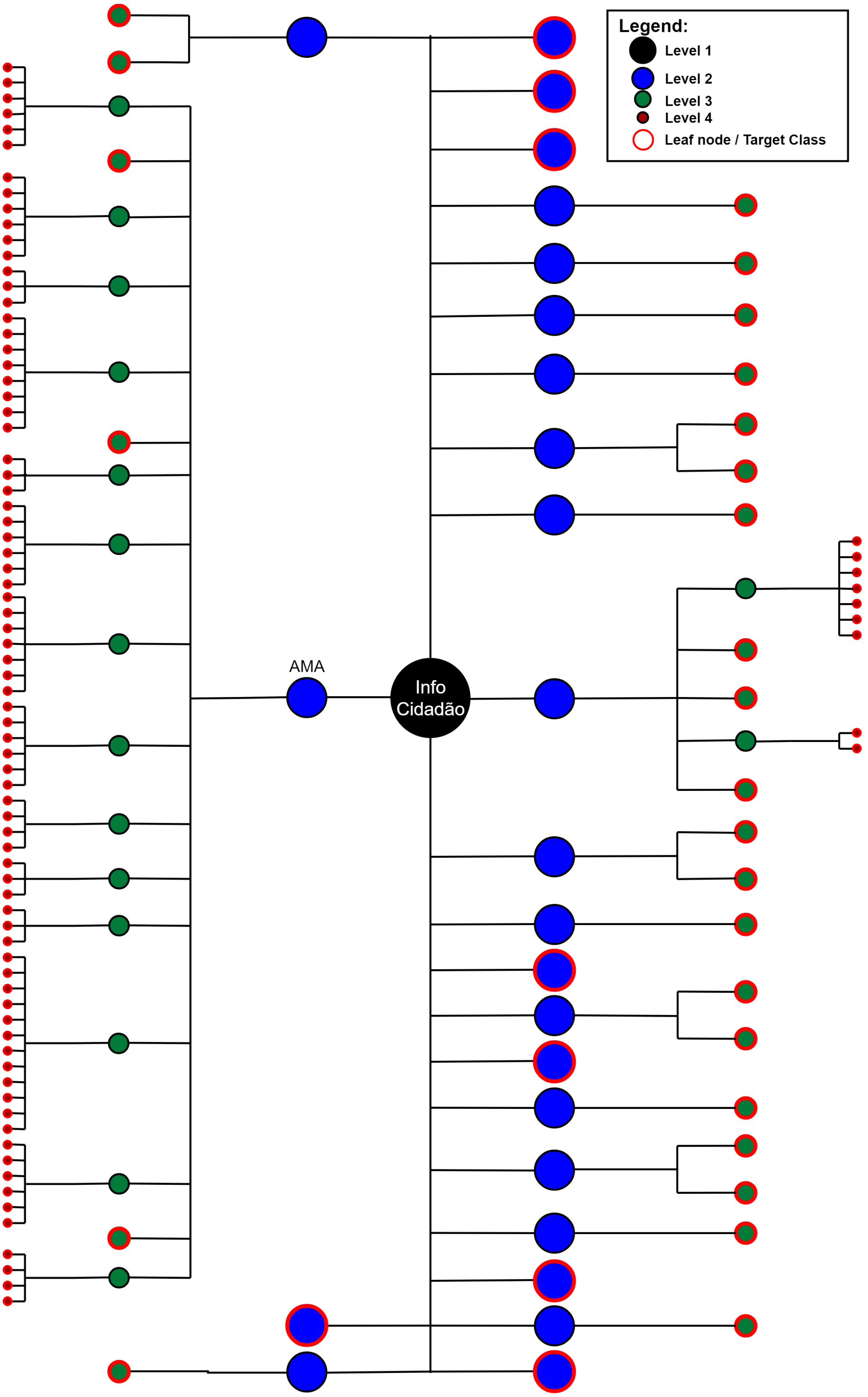


Figure 4.6: Hierarchical Class Structure - Tree Diagram Representation (Av3 dataset)

1. Level 1: has only one class (the root node). It has no meaning, for the classification task, to represent a class level with only one class, but we represent it anyway because its meaning is attached to business rules and because a tree representation must have a root.
2. Level 2: has 25 classes of which 8 are leaf nodes.
3. Level 3: has 42 classes of which 26 are leaf nodes.
4. Level 4: has 86 classes, all leaf nodes.

The target classes for the classification process should only be the leaf node classes, which means that in a flat classification perspective we have 120 classes. There are two problems that can be inferred from this. The first data collection had 102, meaning that at least 18 classes were missing. The second and third data collections had 127 categories and 138 categories, meaning that they had at least 7 and 18 classes, respectively, that were not leaf nodes. Due to the temporal property of the data, our collected data may not have enough samples to portray all the classes available in the business process. Additionally, for business reasons or due to manual classification issues, not all the data collected was classified to leaf node classes. This issue and the solutions applied are explained in section 5.2.

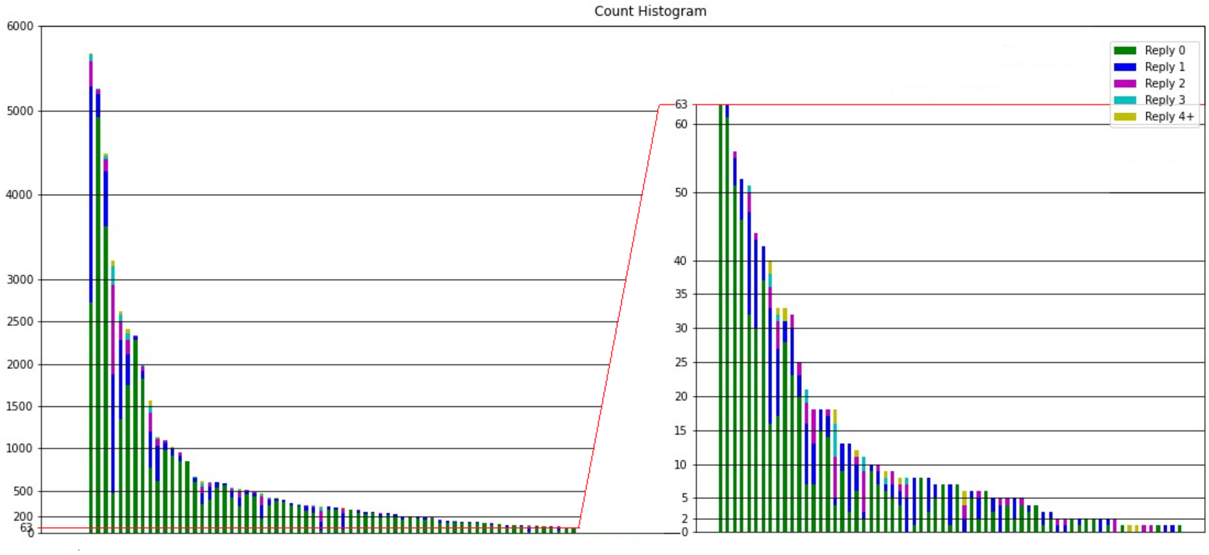


Figure 4.7: Class - email count distribution (red line represents the median statistic of the class email count distribution)

The second data quality problem identified was class imbalance. This problem remained unaltered over the three iterations. Figure 4.7 illustrate the class email count distribution at the third iteration.

Figure 4.8 is an excerpt from the class structure represented in Figure 4.6 which portrays the entire tree. Inter-level nodes with classes assigned (represented with the orange circle) are problematic, because as they have child-nodes, and, semantically, they represent a generalization of their child-node classes, their features are shared by all their child-nodes classes, making it very hard to discriminate between child and parent classes when performing classification.

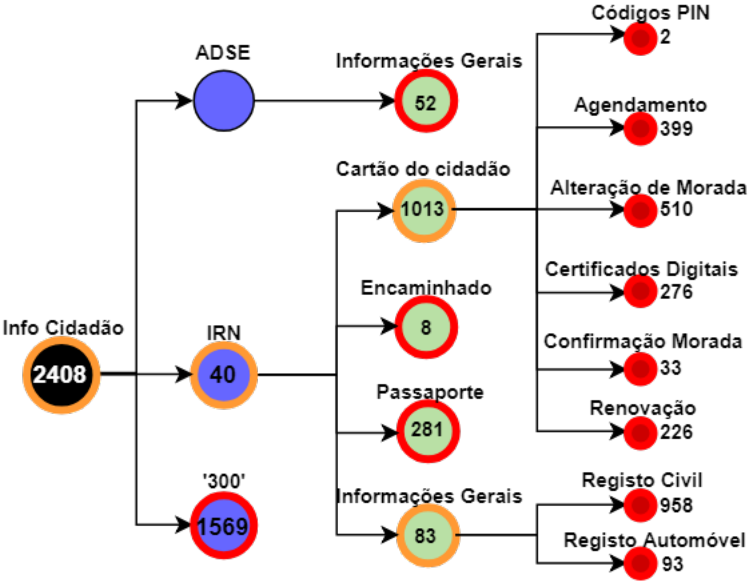


Figure 4.8: Excerpt of the class taxonomy including the files frequency at each class (red are expected target classes and orange are incorrect target classes)

4.3 Summary

The process of data exploration and cleaning concentrates most of the effort dedicated to this project. From understanding the business process to understanding extremely noisy data, the conclusion is that our data is all but trivial. Data cleaning is a continuous process that started when the first raw data was received and still has much work to be done after Av3.

From iteration to iteration, data quantity and data quality was continuously improved. The amount of data increased almost 5 times from the first iteration to the last and an average email in the last (third) iteration has approximately 75% less data than an email at the Av1_raw dataset (first iteration) that was considered noise. The successive evolution in the normalization process and cleaning and normalization actions, enabled the identification of more sentences per email and the identification of more Part-of-Speech (POS) tags per email.

The class taxonomy reveals some semantic and structural problems and the class email distribution portrays a serious case of class imbalance. The structural problems are based on the irregular structure of the tree, where leaf-nodes are located at different levels, and some branches are very wide while other are thin. Furthermore, the fact that some inter-level node classes have emails assigned represents an error to the single-label classification approaches we propose, since due to the transitivity property of the hierarchy tree (see section 2.2.5) a child class inherits all the characteristics of its parents.

5

Features and Models

There is no algorithm for creativity.

– Andy Hargreaves, renowned professor and scholar

Contents

5.1 Design Overview

During the development of this work, a broad range of algorithms, techniques and approaches were explored in order to find an optimal solution to model an email classification system. Figure 5.1 illustrates in a superficial manner the classification system workflows that were experimented. Due to the extensive number of the algorithms, techniques and approaches used, they are shortly described in this chapter, but the results and the most important insights obtained from them are carefully analyzed in the

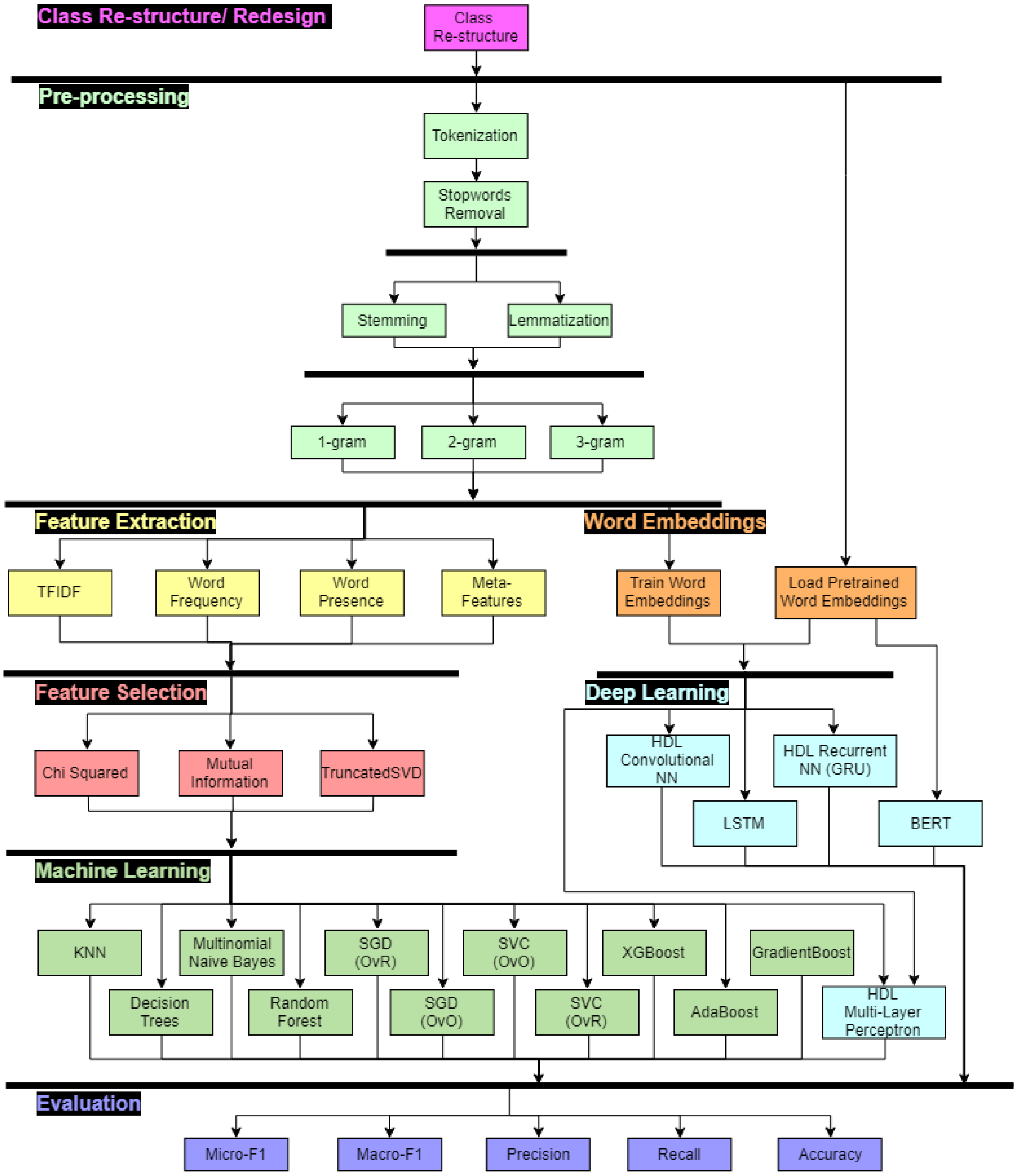


Figure 5.1: Workflow Design

Evaluation chapter (Chapter 6). Independently of the approaches experimented, all the prototypes that were developed aimed to integrate the business process as described in Figure 5.2.

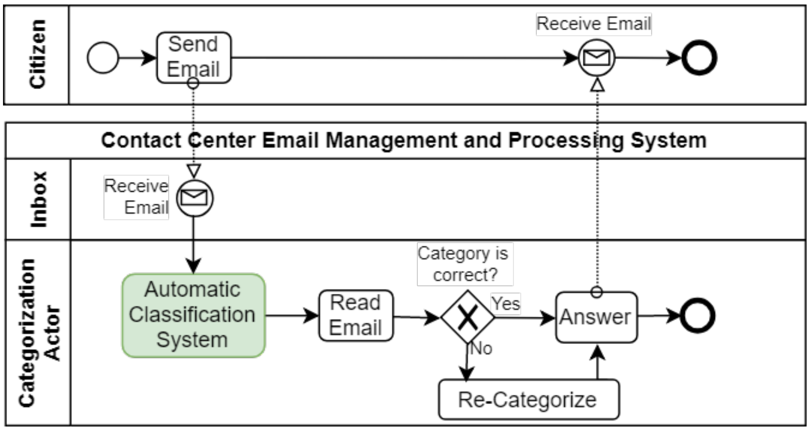


Figure 5.2: Proposed Automatic Email Classification System Integrated in the Contact Center Email Processing Workflow

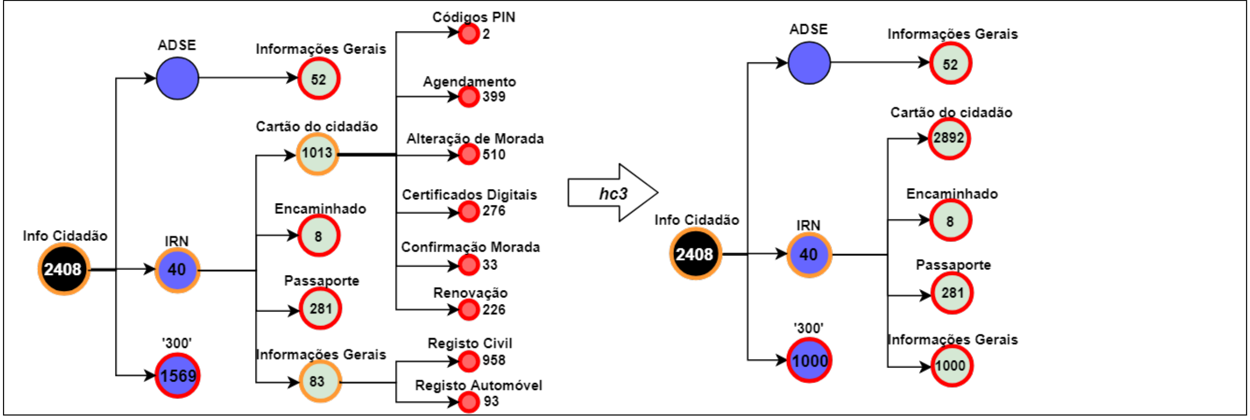
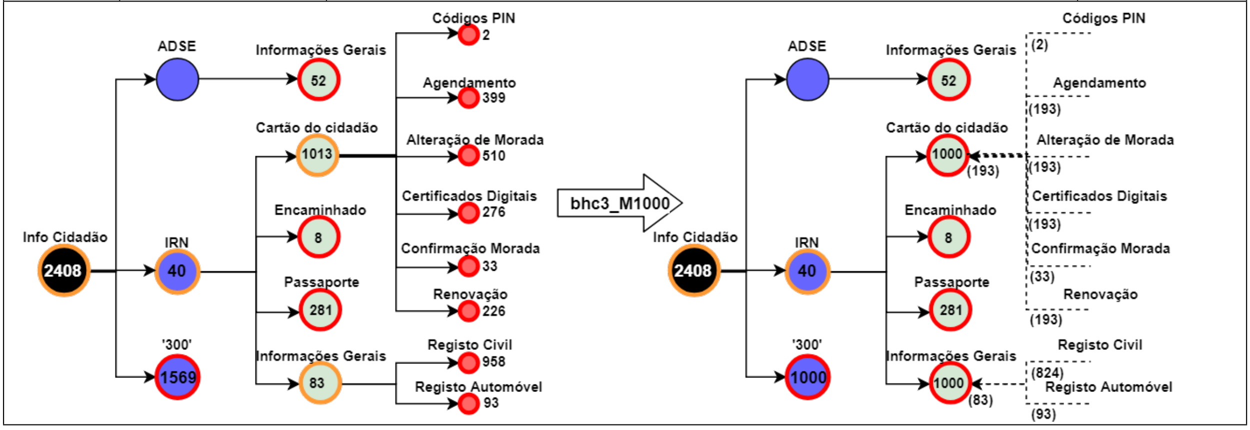
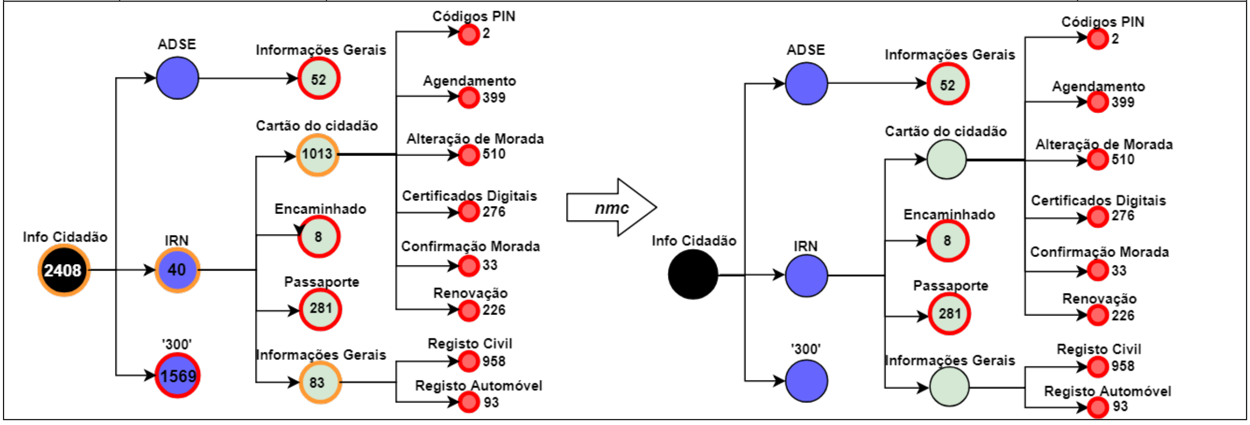
5.2 Class Structure Inspection and Re-structuring

Our dataset, as may be observed in section 4.2.5, has a significant class imbalance problem. The top 3 classes have more than 4000 emails each, while there is 49 classes with less than 10 emails. If our dataset was perfectly balanced with data instances equally distributed between all classes, it would have 49564 / 138 ≈ 359 emails in each category. The reality is that there are only 24 (17,4%) classes with more than 359 emails and 114 (82,6%) classes with less than 359 emails.

Traditionally, the most common class imbalance strategies used are data-level, namely resampling methods. Our dataset has a multiminority problem. Adopting the mean (359 emails per class) as the target, an oversampling strategy involves replicating or synthesizing data instances of 114 classes, where each class would be composed for an average of 23% real data and 77% of synthesized data. On the other side, undersampling the majority classes means undersampling 24 classes and removing an average of 79% of data instances, for each of those classes. Furthermore, oversampling carries the risk of overgeneralization and class overlap, while undersampling risks removing valuable information from each class.

Having this in mind and taking advantage of the hierarchical structure of the classes, the following methods were developed to scrutinize, dissect, and analyse the class set:

1. Hierarchical Cut (Figure 5.1, CERS1);
2. Balanced Hierarchical Cut (Figure 5.1, CERS2);
3. No Middle (inter-level) Class instances (Figure 5.1, CERS3);

Strategy	Name	Rationale	Terminology
CERS1	Hierarchical Cut	Reduce all classes to level L of the hierarchy. All classes that are at a higher level than L are aggregated at L level.	hc$L$
 Info Cidadão 2408 ADSE IRN 40 '300' 1569 Informações Gerais 52 Cartão do cidadão 1013 Encaminhado 8 Passaporte 281 Informações Gerais 83 Códigos PIN 2 Agendamento 399 Alteração de Morada 510 Certificados Digitais 276 Confirmação Morada 33 Renovação 226 Registo Civil 958 Registo Automóvel 93 hc3 Info Cidadão 2408 ADSE IRN 40 '300' 1000 Informações Gerais 52 Cartão do cidadão 2892 Encaminhado 8 Passaporte 281 Informações Gerais 1000			
CERS2	Balanced Hierarchical Cut	Reduce all classes to level L of the hierarchy. All classes that are at a higher level than L are aggregated at L level. It aims to take an equal contribution of instances from all child classes.	bhc$L$_Mmax
 Info Cidadão 2408 ADSE IRN 40 '300' 1569 Informações Gerais 52 Cartão do cidadão 1013 Encaminhado 8 Passaporte 281 Informações Gerais 83 Códigos PIN 2 Agendamento 399 Alteração de Morada 510 Certificados Digitais 276 Confirmação Morada 33 Renovação 226 Registo Civil 958 Registo Automóvel 93 bhc3_M1000 Info Cidadão 2408 ADSE IRN 40 '300' 1000 Informações Gerais 52 Cartão do cidadão 1000 (193) Encaminhado 8 Passaporte 281 Informações Gerais 1000 (83) Códigos PIN (2) Agendamento (193) Alteração de Morada (193) Certificados Digitais (193) Confirmação Morada (33) Renovação (193) Registo Civil (824) Registo Automóvel (93)			
CERS3	No Middle (inter-level) Class instances	Removes classes that are not leaf nodes of the tree but have data instances assigned with them. The aim is to learn to classify emails to the leaf-node classes, so our models cannot learn from these.	nmc
 Info Cidadão 2408 ADSE IRN 40 '300' 1569 Informações Gerais 52 Cartão do cidadão 1013 Encaminhado 8 Passaporte 281 Informações Gerais 83 Códigos PIN 2 Agendamento 399 Alteração de Morada 510 Certificados Digitais 276 Confirmação Morada 33 Renovação 226 Registo Civil 958 Registo Automóvel 93 nmc Info Cidadão ADSE IRN '300' Informações Gerais 52 Cartão do cidadão Encaminhado 8 Passaporte 281 Informações Gerais Códigos PIN 2 Agendamento 399 Alteração de Morada 510 Certificados Digitais 276 Confirmação Morada 33 Renovação 226 Registo Civil 958 Registo Automóvel 93			

Continues on next page

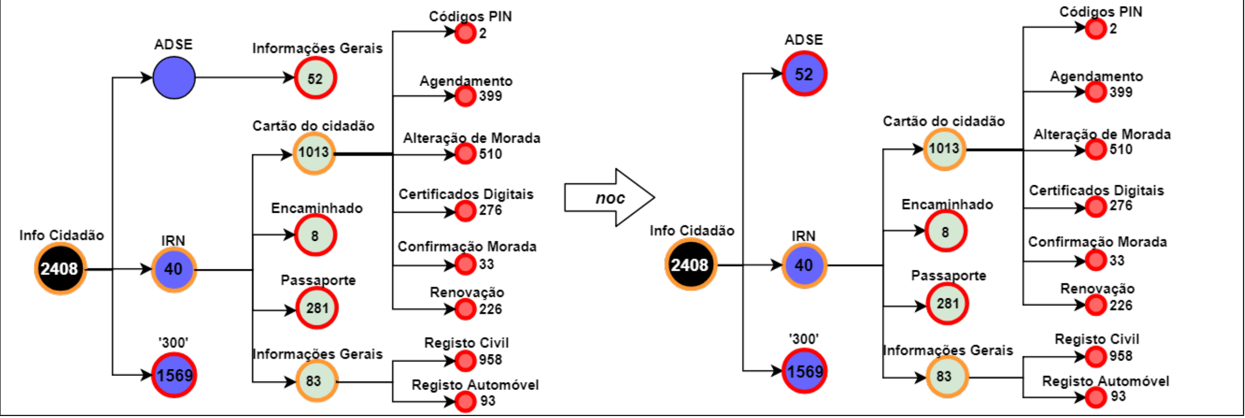
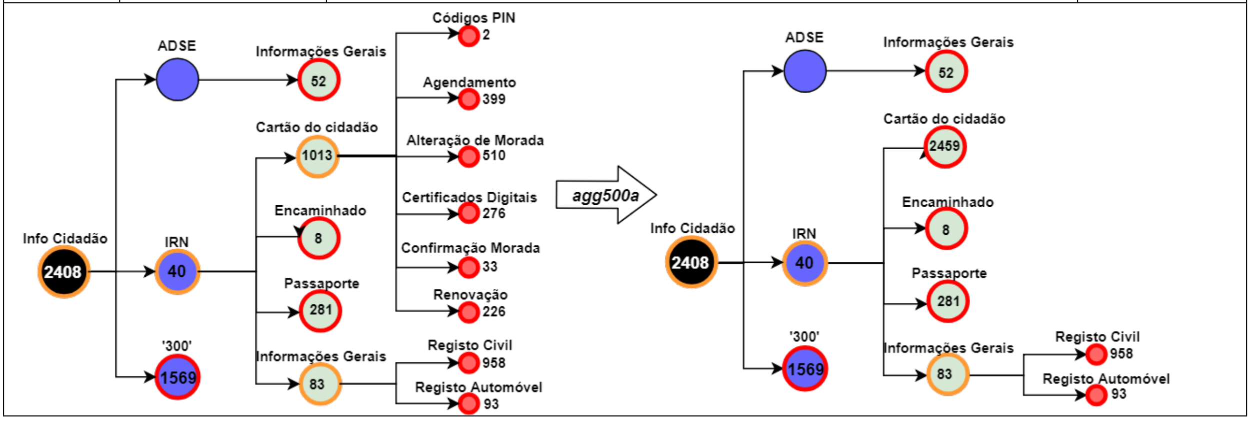
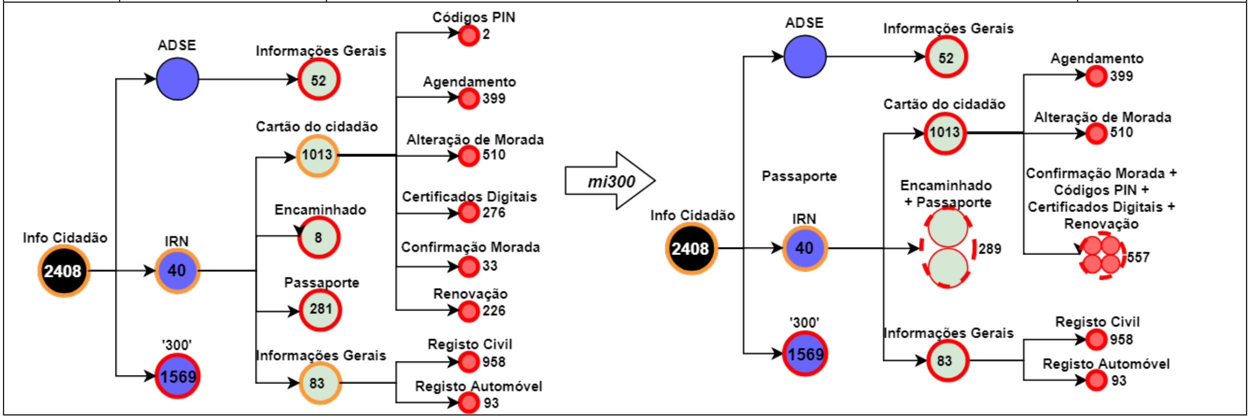
<table>
<tr><th>Strategy</th><th>Name</th><th>Rationale</th><th>Terminology</th></tr>
<tr><td>CERS4</td><td>No Only Child classes</td><td>Aggregates classes of leaf nodes that don't have siblings in the upper class (parent-node). It aims to reduce the depth of the tree at some branches and thus its complexity.</td><td>noc</td></tr>
<tr><td colspan="4">
</td></tr>
<tr><td>CERS5</td><td>Aggregate sibling classes</td><td>Aggregates all siblings leaf-node classes at parent level if they agree to a certain heuristic Lx (x identified the heuristic to be used and L is a value given for the heuristic calculation). The heuristics aim to merge overlapping classes or to reduce class imbalance by merging siblings with a multi-minority problem.</td><td>aggLx</td></tr>
<tr><td colspan="4">
</td></tr>
<tr><td>CERS6</td><td>Merge and Integrate classes</td><td>Aggregates classes that are siblings at their level, under one new class that represents all their documents and is named by the concatenation of their names.</td><td>mi</td></tr>
<tr><td colspan="4">
</td></tr>
</table>

Table 5.1: Strategies for Exploring and Re-structuring the Class Structure (numbers represents the amount of emails assigned to the class).

4. No Only Child (Figure 5.1, CERS4);
5. Aggregate sibling classes in parent category (Figure 5.1, CERS5);
6. Merge classes (Figure 5.1, CERS6);
7. Remove classes with less than *n* instances or more than *N* instances;
8. Undersample;
9. Manual selection of classes to be removed or merged.

These approaches were not used single-handedly but in conjunction. The most common combinations are described in appendix B.2, in Table (B.2), where several relations between the combinations of strategies with data distribution and class frequency statistics may be analyzed. These combinations are built to meet specific goals, namely to perform hierarchical classification, to reduce class imbalance, class overlap, the impact of label noise, but ultimately to increase classification performance.

5.2.1 Class Structure Redesign

The complexity of our data class structure is shown in Figure 4.6. It does not show each class data instances count, but despite we represented our tree, regarding classification, as a virtual tree (where only the leaf nodes are the classification target classes) [Silla and Freitas, 2011], our data included some instances labelled with inter-level classes (13.3% of data instances labeled within 26 inter-level classes). We regarded inter-level classes as a major cause for low classification performance, so the No Middle Class (CERS3 of Table 5.1) instances strategy comes to solve this problem by removing these classes from the target classes set and their data instances. Note that if we do a hierarchical cut in level two for example, these inter-level classes may become a leaf-node class in which case their data instances will be aggregated in leaf-node classes and not removed.

Figure 4.6 also shows some nodes that have only one child node. Having only one child node does not present a classification challenge when going from a node to the next level node, but it adds to the class structure complexity. In order to reduce complexity, No Only Child classes strategy (CERS4 of Table 5.1) aggregates only child node class instances in the parent class and removes the only child node class from the classification class target set.

5.2.2 Hierarchical Classification

In order to perform hierarchical classification in a hierarchy level-based approach, hierarchical cut strategies extract the classes at a given level. For example, to learn to classify our Av3 dataset to the leaf-node using a top-down hierarchical approach, with level-based classifiers, we "divide" the datasets in three offspring datasets by applying a Hierarchical Cut (CERS1 of Table 5.1) or Balanced Hierarchical Cut

(CERS2 of Table 5.1) strategy on Av3: Av3_bhc2, Av3_bhc3, Av3_bhc4 (in case of balanced hierarchical cut). Then we train multiple models for each of these datasets and select the model with the best performance for each offspring dataset. The hierarchical classification model for Av3 is the composition of Av3_bhc2, Av3_bhc3, Av3_bhc4 models.

5.2.3 Class Re-balance

Both in hierarchical classification and in flat classification, our datasets present a huge class imbalance problem. Since we have a multi-minority problem, we decided not to use oversampling strategies in order to maintain data fidelity, otherwise, we would have a dataset where the majority of data was synthesized. Undersample strategies present the downfall of removing a huge fraction of our data. Nevertheless, undersample strategies were adopted in some cases such as the classification of certain class levels ('hierarchically cut' datasets) mainly due to the the fact that other 'class imbalanced concerned' strategies did not achieved the desired performance.

Some classes are extremely ill represented. Figure 4.7 shows that there are 61 classes with 25 or less data instances. 25 data instances are 0.05% of data, which is almost meaningless, and there are 9 classes with only 1 data instance (0.002% of data), which cannot be used in the classification process, since we cannot divide these classes instances in training and test sets.

To deal with these extreme situations our strategies fall into two ways of thinking:

1. Delete class with an extremely low percentage of data instances. A class with 25 data instances represents only 0.05% of data, which is almost insignificant but may have a negative influence over the classification decision of the other 99.95% of data and originate a considerable impact on the performance results. So removing these classes and their data instances from the data submitted to the learning process may led to better classification models. We employed a strategy to remove classes with less than a minimum of instances *N* ('Remove classes with less than *N* instances').
2. Merge class with an extremely low percentage of data instances. This project intends to have an application in the real-world, so we do not want to loose any data, as it may be important in the real-world processes where it fits into. The alternative is to group the low percentage data instances in one or more classes in order for them to be more representative of the data. We employed two main strategies to merge classes:
    - The Merge classes strategy (CERS6 of Table 5.1) merges sibling classes if they agree to an heuristic (the heuristic most applied takes into consideration the sibling class proportions and a given value).
    - The Aggregate sibling classes (CERS5 of Table 5.1) in parent node category aggregates all siblings of a class in the parent-node class.

5.2.4 Partial Class Set Classification and Manual Selection

Hierarchical Classification may be performed by branch, where models are trained for selected branches instead of for each level. Strategies to manually selecting a class and its children or to exclude certain classes from the model ('Remove classes with more than *N* instances') were developed to support the training of models to specific subsets of the dataset. Furthermore, strategies to merge only selected classes were developed, which have multiple applications, from merging carefully selected sibling nodes to merging all the nodes in a certain level in two new nodes, creating an artificial level of aggregated nodes.

5.3 Text Pre-processing

The Portuguese language is a highly inflectional language, so the recognition of morphological variation and conceptual proximity of the words is a crucial task. The most common approaches to lexical normalization, stemming and lemmatization, were both applied to our data and their performance was compared in a multitude of tests combining different feature extraction methods, feature selection methods and classifiers.

The stemming algorithms applied were Porter [Porter, 1980] and RSLP [Moreira and Huyck, 2001], implementations from the NLTK[1] library, and the lemmatization was done using *spaCy*'s lemmatizer (Portuguese large model "pt_core_news_lg"[2]). These stemming algorithms were chosen due to their popularity and the performance results with the Portuguese language reported at [Flores and Moreira, 2016].

During this stage, tokenization usually preceded stemming or lemmatization algorithms, where texts were transformed into lists of tokens, and stop-word removal was also applied, where a set of predefined stop-words for the Portuguese language, from NLTK python library, was used.

5.4 Metadata and Custom Features

Text classification usually gets its features from text. However, the quality of features extracted from text may not be enough for the ML models to achieve satisfactory results.

Feature engineering, namely the use of metadata and the creation of custom features are common approaches to have features with high information gain in order to improve the performance of ML algorithms.

We used four approaches to create custom features:

[1] https://www.nltk.org/

[2] https://spacy.io/models/ptpt_core_news_lg

1. Reply number:

   Reply number was a custom feature created at the data understanding phase (at the task of building a dataset from raw data). It represents the position of the email at the ordered chain of emails;

2. Text statistics:

   For each text, we extracted the sentence count, word count, unique words count, character count, average character per word, average words per sentence, verb count, unique verb count, verb percentage (in relation to word count), root verb count, unique root verb count and unique root verb percentage (in relation to word count). We also extracted statistics from named entities and POS tags, but the observed results from the NER and POS tagging processes were not good enough and so we decided to drop them;

3. Previous email category:

   The subject of an email chain is usually constant and the category attributed to an email is usually related with the subject of the email. Naturally, it is deductible that the previous class is a relevant feature, so for each email where "Reply number" $>$ 0, the name of the previous category was extracted;

4. Categories keywords:

   Emails are classified with categories which name represents the nature/subject of the email. There is a big probability that the email contains in its body the same keywords that describe the categories. For each level of the category hierarchy structure, the keywords that make the name of each category were identified. For each email, these keywords are counted using a fuzzy match algorithm [3] (to account for misspellings and noise) that matches any word that has more than 80% similarity score (using Levenshtein distance).

5.5 Feature Extraction

Introduced in section 2.4 and described in appendix C.2, there are numerous feature extraction algorithms. However, from the research that was done and presented, TF-IDF tends to get the best performance (chapter 3).

In our work, the feature extraction method that was more frequently used during the development of our email classification system was TF-IDF[4]. However, other feature extraction methods were applied, such as TF[5], which converts a collection of text documents to a vector of term (token) counts. Using this algorithm we also converted our collection of emails to Term Presence (1 if the word as at least one

[3] https://pypi.org/project/fuzzywuzzy/

[4] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

occurrence and 0 otherwise).

A brief example of the impact of the feature extraction techniques is shown in Table 6.3 (in the next chapter).

5.6 Feature Selection

From the work related to email/text multiclass classification, we see a tendency to the feature selection algorithms of Chi-squared and Mutual Information (MI) / Information Gain (IG)[6]. Following the performances reported in multiple works (described in chapter 3), we decided to experiment with them. Furthermore, in search of better performance, we experimented with *TruncatedSVD*[7], a dimensionality reduction algorithm that performs singular value decomposition and is presented in apendix C.3. Comparative results are shown in Figure 6.5 (in the next chapter).

5.7 Word Embeddings

DL algorithms, usually get the best performances when combined with word embedding features, mentioned in section 2.4 and appendix C.1. During the development of this work, several DL algorithms were experimented, where both locally trained word embedding models and pre-trained models were used.

When training our own word embedding models, during the second and third iteration of the CRISP-DM methodology, we used combinations of dimension sizes (100 and 300) with window sizes (3 and 5). To train these models we used python the library *gensim*[8].

5.8 Classification

The objective of this work is to classify emails into categories. The classifiers will be trained to create models aiming to get the best performance. With the purpose of achieving the best performance, several algorithms were implemented, from simple algorithms such as the traditional Machine Learning (ML) techniques to the more complex Deep Learning (DL) models. Table 5.2 shows the algorithms used.

A huge fraction of the algorithms employed used the scikit-learn library implementation. These algorithms are briefly presented in appendix C.4. Most of these algorithms used the default parameter settings. With K-Nearest Neighbors (KNN) we used k=3 most of the time but hyperparameter optimization tests using the grid search method showed that 3 was not the best value of k, that the best values of

[6] According to the references in the web page (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html) of the MI algorithm we used , MI and IG are the same thing

[7] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

[8] https://radimrehurek.com/gensim/

k are located between 20 and 50 and that the best performing k value changes with the pre-processing steps taken, namely the class re-structure strategies and the feature extraction and feature selection methods.

Gradient Boosting and AdaBoost obtained good performance scores, but were always behind XGBoost. Because of this and because they are very computationally expensive they were only used with Av2, were they were first introduced.

XGBoost (eXtreme Gradient Boosting) obtained top performance scores. Because of this it was the target of hyperparameter tuning techniques in order to increase its performance even more. Since XGBoost is very computationally expensive and in order to optimize its hyper-parameters we decided not to use *XGBClassifier* (Scikit-Learn Wrapper interface for XGBoost) and instead build a custom classifier from XGBoost Booster object. Our data was represented in our project with an object (*Dataset*) that followed the Natural Language ToolKit (NLTK) datasets interface for ease of use and that we coded aiming for adaptability to multivariate feature engineering methods and for portability to diverse classification approaches, but not aiming for efficiency (data was represented in a DataFrame attribute of the object). In order to increase efficiency our custom XGBoost object converted our data to the XGBoost native data format, DMatrix. The training was performed using XGBoost.train() method with early stopping (target: maximize macro-F1-measure, patience: 100 epochs). The parameters used were found either by the use of hyperparameter optimization methods (RandomSearchCV[9]), as was the case of the learning rate (0.01) and max depth of a tree (6), selected for computational efficiency (GPU predictor and 'gpu_hist' tree method) or found empirically. The training process returns a boosted random forest model from the best epoch. The custom built XGBoost was found to have a better performance by a margin of 5-7% points over accuracy and macro-F1-measure than the predefined scikit-learn XGBoost wrapper.

The HDLTex (Hierarchical Deep Learning) classifiers (DNN, CNN and RNN) are described by [Kowsari et al., 2017]. The implementation of HDLTex is available at github[10]. We studied the HDLTex implementation, which was adapted to a simpler classification problem: text classification of WOS datasets, which are text classification benchmark datasets with a class structure organized in two levels, where the first level represents a major subject and the second level represents minor subjects and contains all the leaf-nodes. We extracted the code used to build the three classifiers described by [Kowsari et al., 2017] and adapted it to our project. HDLTex CNN suffered some considerable changes in its adaptation: it uses the same layers, but their complexity was reduced because of data constraints.

RMDL is and "a new ensemble, deep learning approach for classification" by [Kowsari et al., 2018, Heidarysafa et al., 2018], available at github[11], which solves the problem of finding the best deep learn-

[9] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[10] https://github.com/kk7nc/HDLTex

[11] https://github.com/kk7nc/RMDL

Classifier	Implementation	Hyper-parameters
Multinomial Naïve Bayes (MNB)	Scikit-learn[a]	(default)
Decision Tree Classifier (DT)	Scikit-learn[b]	(default)
K-Nearest Neighbors (KNN)	Scikit-learn[c]	k = 3 by default and all odd k values between 3 and 55 on some experiments
Support Vector Classifier (SVC) One vs Rest (One vs All) (OvR)	Scikit-learn[d]	(default)
Support Vector Classifier (SVC One vs One (OvO))	Scikit-learn[e]	(default)
Stochastic Gradient Descent (SGD) One vs Rest (OvR)	Scikit-learn[f]	(default)
Stochastic Gradient Descent (SGD) One vs One (OvO)	Scikit-learn[g]	(default)
Random Forest Classifier	Scikit-learn[h]	(default)
GradientBoosting	Scikit-learn[i]	(default)
AdaBoost	Scikit-learn[j]	(default)
XGBoost Classifier[k]	Custom using XGBoost booster	Optimized with Random Search, best model chosen using early stopping
HDLTex DNN (Multi Layer Perceptron)	HDL github[l]	Best model chosen using early stopping
HDLTex CNN (Convolutional Neural Network)	HDL github[m]	5 levels of complexity, best model chosen using early stopping
HDLTex RNN (Recurrent Neural Network using GRU)	HDL github[n]	Best model chosen using early stopping
Random Model Deep Learning (RMDL)	RMDL github[o]	Experimental
LSTM	Tensorflow	Experimental
BERT	pyTorch	Experimental

Table 5.2: Classifiers Used

[a] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
[b] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[c] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html
[d] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[e] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[f] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[g] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[h] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[i] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[j] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
[k] https://xgboost.readthedocs.io/en/latest/index.html
[l] https://github.com/kk7nc/HDLTex
[m] https://github.com/kk7nc/HDLTex
[n] https://github.com/kk7nc/HDLTex
[o] https://github.com/kk7nc/RMDL

ing structure. It randomly searches the best hyperparameters for three different DL architectures: MLP, RNN, CNN. This algorithm was scarcely used when compared to the others, since it is very computationally expensive.

BERT (Bidirectional Encoder Representations from Transformers) was proposed in [Devlin et al., 2018]. BERT is based on a multi-layer bidirectional Transformer [Vaswani et al., 2017] and uses a Masked Language Model to predict words which are randomly masked or replaced. BERT is the first fine-tuning-based representation model that achieves state-ofthe- art results for a range of NLP tasks. “BERT-base model contains an encoder with 12 Transformer blocks, 12 self-attention heads, and the hidden size of 768. BERT takes an input of a sequence of no more than 512 tokens and outputs the representation of the sequence. The sequence has one or two segments that the first token of the sequence is always [CLS] which contains the special classification embedding and another special token [SEP] is used for separating segments. For text classification tasks, BERT takes the final hidden state h of the first token [CLS] as the representation of the whole sequence. A simple softmax classifier is added to the top of BERT to predict the probability of label c: $p(cjh) = softmax(Wh)$ where W is the task-specific parameter matrix” [Chi et al., 2019].

Our BERT classifier is built on top of “NeuralMind BERT-base Portuguese Cased” model[12] by [Souza et al., 2020]. The optimizer used is AdamW [Kingma and Ba, 2017]. All hyper-parameters were defined experimentally.

5.9 Development Overview

All the approaches presented in this chapter were not planned or developed at once, in the beginning of the project, but the product of our development methodology, where at each iteration needs were identified and further algorithms were planned to the following iteration, in order to answer the limitations or lack of performance identified. Figure 5.3 shows a succinct overview of the development, where the major decisions and actions (but not all) taken at each iteration/phase are presented.

Altogether, these experiments produced a large number of results, resulting in an extensive report on the use of preprocessing, feature extraction, feature selection, dimensionality reduction, word embedding and ML and DL models to solve the challenges proposed by noisy data organized in a complex hierarchical structure of classes.

[12] https://huggingface.co/neuralmind/bert-base-portuguese-cased

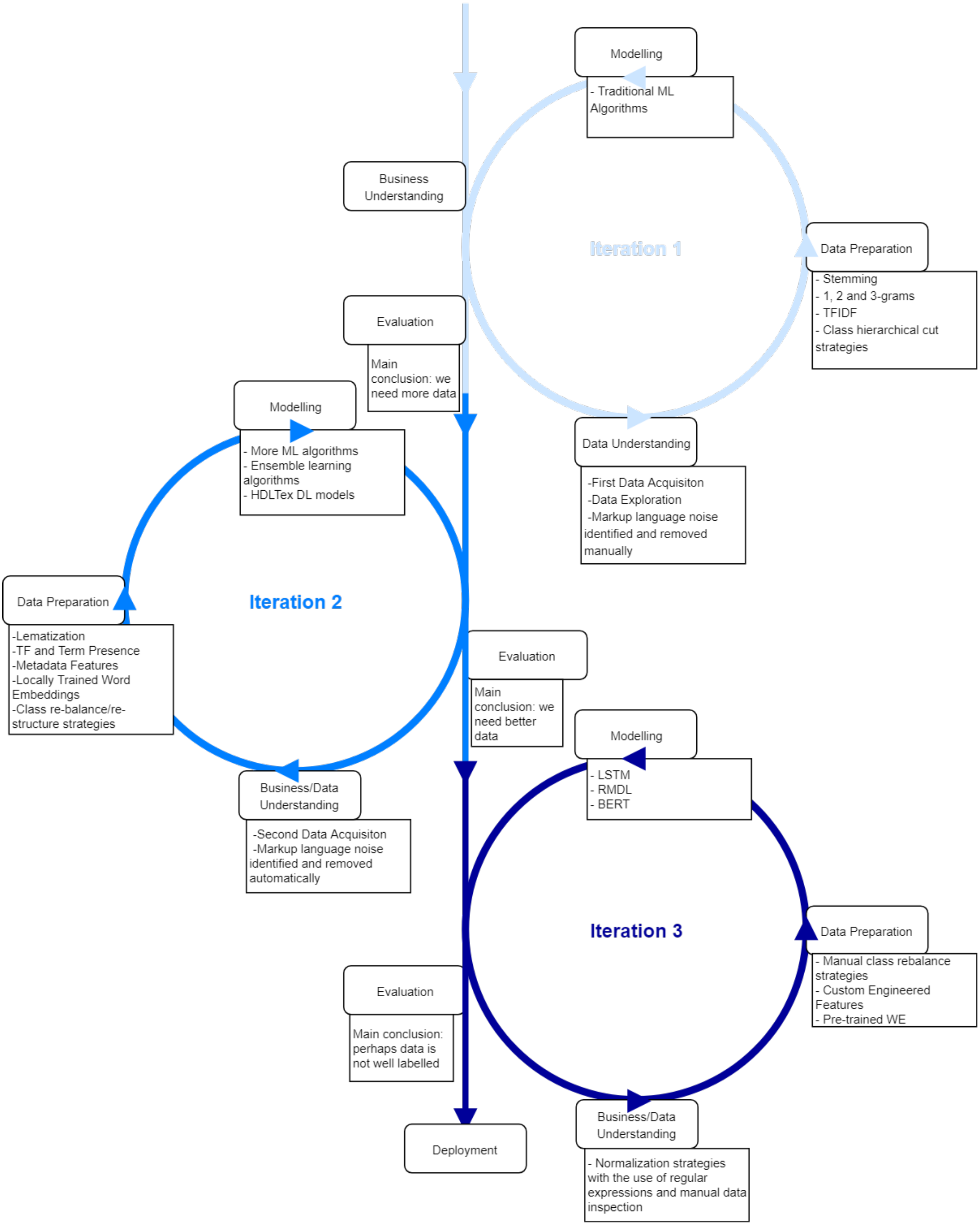

Modelling
- Traditional ML Algorithms
Business Understanding
Iteration 1
Data Preparation
- Stemming
- 1, 2 and 3-grams
- TFIDF
- Class hierarchical cut strategies
Evaluation
Main conclusion: we need more data
Data Understanding
-First Data Acquisiton
-Data Exploration
-Markup language noise identified and removed manually
Modelling
- More ML algorithms
- Ensemble learning algorithms
- HDLTex DL models
Iteration 2
Data Preparation
-Lematization
-TF and Term Presence
-Metadata Features
-Locally Trained Word Embeddings
-Class re-balance/re-structure strategies
Business/Data Understanding
-Second Data Acquisiton
-Markup language noise identified and removed automatically
Evaluation
Main conclusion: we need better data
Modelling
- LSTM
- RMDL
- BERT
Iteration 3
Data Preparation
- Manual class rebalance strategies
- Custom Engineered Features
- Pre-trained WE
Evaluation
Main conclusion: perhaps data is not well labelled
Business/Data Understanding
- Normalization strategies with the use of regular expressions and manual data inspection
Deployment

Figure 5.3: CRISP-DM cycle deconstructed and main actions per phase

6

Evaluation

If things are not failing, you are not innovating enough.

– Elon Musk

Contents

Following the CRISP-DM methodology and development reported in previous chapters, we had three evaluation phases, to evaluate the major datasets Av1, Av2 and Av3. Each of these major datasets was submitted to class re-structure approaches, producing several offspring datasets. Each offspring dataset was submitted to several text preprocessing steps such as n-gram-representation, stemming, lemmatization, meta-features and hybrid combinations of these. Furthermore, each pre-processed dataset was submitted to some of several feature extraction and feature selection methods. Then, each dataset with each data instance represented by its pre-processed selected features was classified by multiple classifiers.

All the possible combinations of different actions at each step produced a multitude of evaluation results. To be able to present the most relevant results, we will present the best results per type of classification strategy (flat classification, local classification per level and local classification per parent node) followed by the relevant insights from the use of our different strategies regarding feature extraction/selection and classifier methods.

6.1 Evaluation Metrics

Let TP, FP, TN, FN denote true positives, false positives, true negatives and false negatives. The most primary metrics to evaluate the quality of a classification model are Accuracy and Error rate (equation 6.1).

$$\text{Accuracy} = \frac{(TP + TN)}{N}, \qquad \text{Error rate} = \frac{(FP + FN)}{N} \tag{6.1}$$

where N is the total number of data instances. Obviously we have $\text{ErrorRate} = 1\text{-Accuracy}$.

Precision, Recall and F-measure (also called F1-Score) are also primary metrics typically used to measure performance for imbalanced datasets. Equation 6.2 defines precision, recall and F1-measure (which is the harmonic mean of the precision and recall) for binary classification.

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \qquad \text{F1-measure} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6.2}$$

For multi-class classification, precision, recall and F1-measure can be calculated for each individual class and averaged to get an overall performance value. The average of class-wise F-measures is given by macro-F1, while a F-measure obtained by the overall precision and recall is called micro-F1 (see Equation 6.3).

$$\text{macro-F1} = \frac{1}{Y} \cdot \sum_{y \in Y} \mathit{F1}(y), \qquad \text{micro-F1} = \frac{2 \cdot \mathit{Precision}_{micro} \cdot \mathit{Recall}_{micro}}{\mathit{Precision}_{micro} + \mathit{Recall}_{micro}}, \tag{6.3}$$

$$\text{where} \quad \text{micro-Precision} = \frac{\sum_{y \in Y} TP_y}{\sum_{y \in Y} TP_y + \sum_{y \in Y} FP_y}, \qquad \text{micro-Recall} = \frac{\sum_{y \in Y} TP_y}{\sum_{y \in Y} TP_y + \sum_{y \in Y} FN_y}$$

,where Y is the set of classes.

The identity of overall micro-Precision and micro-Recall causes their identity with micro-F1, meaning micro-F1 = micro-Recall = micro-Precision. This measure is identical to overall accuracy.

In hierarchical classification, multi-model classification models usually use more complex performance measures. There are no standard measures for complex models [Silla and Freitas, 2011]. In multi-model classification models, we call the local models performance the ‘Local’ performance (eg. Local Accuracy) and the composite model performance the ‘Overall’ performance (eg. Overall Accuracy). ‘Local’ performances are given by the metrics aforementioned. ‘Overall’ performance is given by:

$$\text{Overall Performance} = \prod_{i}^{n} \text{Local Performance}_\text{i} \tag{6.4}$$

, where n is the number of models.

6.2 Flat Classification

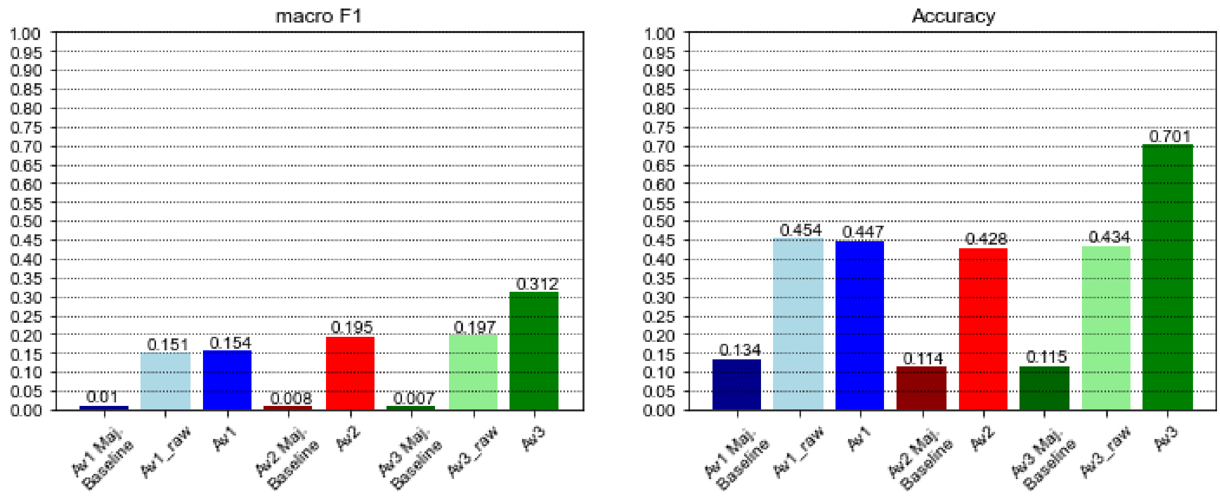
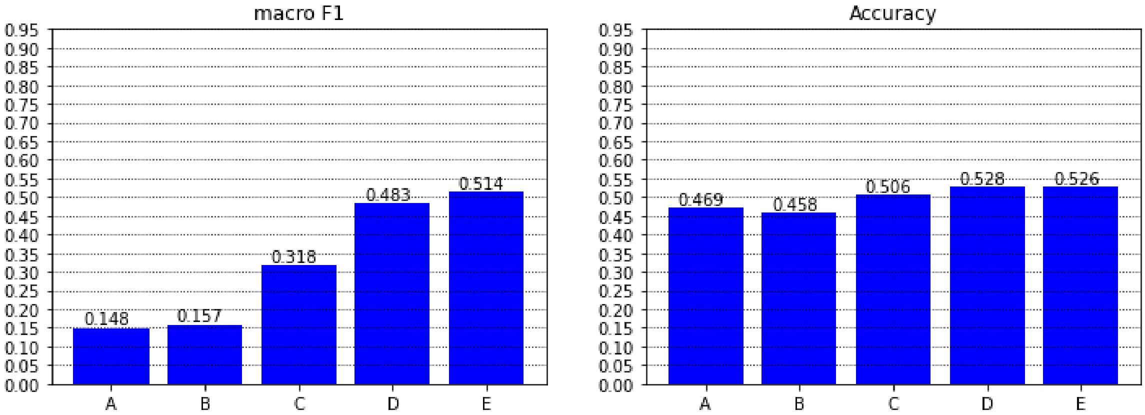


Figure 6.1: Evaluation results (macro-F1-measure and accuracy) of datasets Av1 (blue), Av2 (red) and Av3 (green), using XGBoost and TF-IDF

Flat classification ignores the class relationships and performs prediction considering all leaf nodes as independent classes. Figure 6.1 shows that the results increase from dataset to dataset, which means that increasing data amount and data quality benefits performance.

There are a number of factors that influence accuracy. It usually presents high results in the imbalanced datasets because the classifier is biased towards the majority class. From Av1 to Av3 the proportion of data in the majority class decreased in proportion to the other classes, which may explain the small decrease in accuracy from Av1_raw to Av3_raw and also explains the decrease in majority baseline. Av3 macro F1-measure and accuracy increase approximately 63% and 62% over Av1, which may be explained by the exponential increase in data quality provide by the data normalization and noise removal described in Chapter 4.



	Offspring datasets - Re-balance	**Total Classes**	**Total Emails**	**Classifier**	**Features**
A	Av1	105 (100%)	11894 (100%)	OvO - SGD	TFIDF-3500 1-gram, Chi2-1000
B	Av1_noc	98 (93,3%)	11894 (100%)	OvO - SGD	TFIDF-3500 1-gram, Chi2-1500
C	Av1_noc_ag100a_nmc	40 (38,1%)	9179 (77,2%)	OvO - SGD	TFIDF-3500 1-gram, Chi2-1000
D	Av1_noc_ag100a_nmc_mi100_m100	13 (12,4%)	8570 (72,1%)	OvO - SGD	TFIDF-3500 1-gram, Chi2-250
E	Av1_noc_ag100a_nmc_mi100_m100_u100	13 (12,4%)	1690 (14,2%)	OvR - SVC	TFIDF-3500 1-gram, Chi2-250

Figure 6.2: Evaluation results of Av1 offsprings

Figure 6.2 shows that class re-structure techniques described in section 5.2 improve macro-F1 performance mainly by reducing the number of classes with very low amount of samples by removing them and by aggregating them and creating classes with more significance.

6.3 Local Classification Per Level Approach

The local classification per level approach consists of training one multi-class classifier for each level of the class hierarchy.

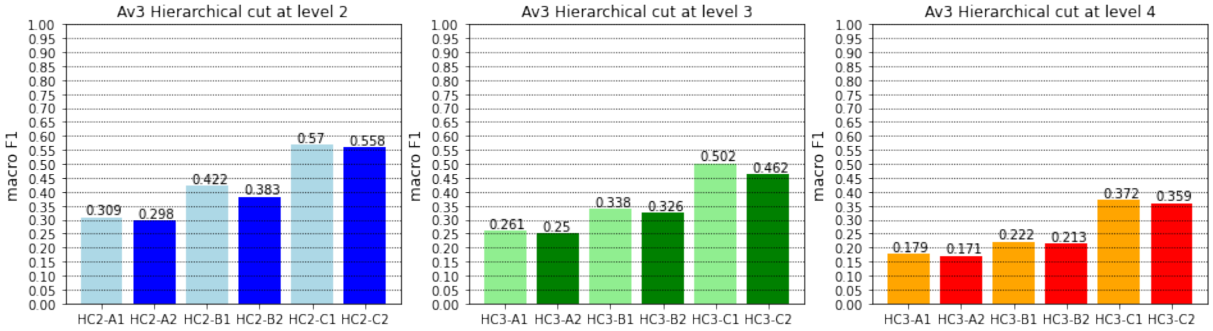


Figure 6.3: Evaluation results (macro-F1-measure and accuracy) of hierarchically cutted datasets

	Offspring datasets - Re-balance	**Total Classes**	**Total Emails**	**Classifier**	**Features**
HC2-A1	Av3_hc2	49564	25	XGBoost	TFIDF-3500 1-gram
HC2-A2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-1500
HC2-B1	Av3_bhc2_M500	6259	24	XGBoost	TFIDF-3500 1-gram
HC2-B2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-500
HC2-C1	Av3_bhc2_M500_mi100_m100	6073	15	XGBoost	TFIDF-3500 1-gram
HC2-C2				OvO - SVC	TFIDF-3500 1-gram, TruncSVD-2500
HC3-A1	Av3_hc3	49564	62	XGBoost	TFIDF-3500 1-gram
HC3-A2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-2000
HC3-B1	Av3_bhc3_M500	11193	49	XGBoost	TFIDF-3500 1-gram
HC3-B2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-2000
HC3-C1	Av3_bhc3_M500_mi100_m100	10837	29	XGBoost	TFIDF-3500 1-gram
HC3-C2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-500
HC4-A1	Av3_hc4	49564	126	XGBoost	TFIDF-3500 1-gram
HC4-A2				OvR - SVC	TFIDF-3500 1-gram, TruncSVD-1500
HC4-B1	Av3_bhc4_M500	17788	104	XGBoost	TFIDF-3500 1-gram
HC4-B2				OvO - SVC	TFIDF-3500 1-gram, TruncSVD-1500
HC4-C1	Av3_bhc4_M500_mi100_m100	17149	51	XGBoost	TFIDF-3500 1-gram
HC4-C2				OvO - SVC	TFIDF-3500 1-gram, TruncSVD-1000

Table 6.1: “Hierarchically cut” offspring datasets statistics

We use the strategies of hierarchical cut and balanced hierarchical cut, presented in section 5.2 to obtain level-class-based datasets to train each level classification models. Figure 6.3 (and complementary Table 6.1) show that just by aggregating ”cutted” classes with balance, using Balanced Hierarchical Cut (Figure 5.1) macro-F1-measure increases, probably due to the fact that the new generated dataset classes have less class imbalance and are more cohesive.

By further removing ill represented classes (classes with very low amount of samples) we can also observe that macro-F1-measure increases substantially, with the sacrifice of only a very small amount of data.

These observations act like a rule, for all cut levels. Macro-F1-measure reduces from level to level because the amount of classes increase and because, as explained in section 2.2.5 and Chapter 4, the boundary between classes is smaller which makes class overlap occur more dramatically. Since lower level classes are more hard to distinguish, they are more susceptible to label noise. These factors combined (number of classes, class overlap and label noise) may be the cause for the continuous decrease of performance from "hierarchically cut" level to "hierarchically cut" level.

Despite we call these approaches as local classification per level, we do not in fact create a multi-model classification model, composed of a model trained for level 2, a model trained for level 3 and a model trained for level 4. Traditionally, local classification per level is useful when the class path is important, which are the situations when a given node may have more than one parent node (DAG [Silla and Freitas, 2011]). Since we have a tree, our local classification per level approaches are more of flat classification approaches where classes have been "hierarchically cut". For example dataset Av3_hc3 includes not only leaf nodes at level 3, but also leaf_nodes at level 2 that did not had any child nodes originally. Furthermore, if we used Equation 6.4 to compute an overall performance of a multi-model composed by local models Av3_bhc2_M500_mi100_m100 (level 2) and Av3_bhc3_M500_mi100_m100 (level 3), the overall macro F1 would be about 28.6% ($0.57 \cdot 0.502$) in the best case scenario.

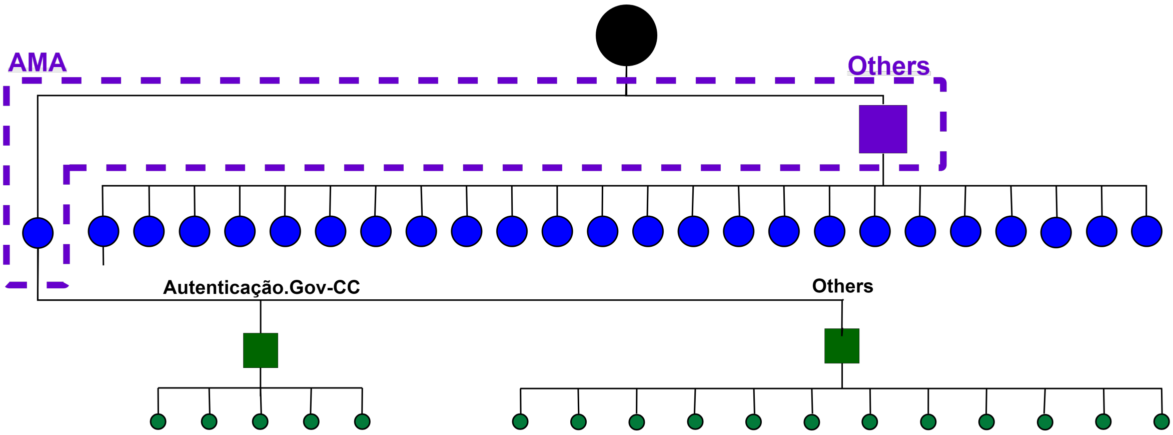
6.4 Local Classification Per Parent Node-based Approach

Classically, a local classification per parent node approach is a top-down approach where we train a model to classify each parent-node's children At the second level of our hierarchy, this strategy does not differ from the local classification per level approach, because all nodes at this level are siblings.

In order to further explore the hierarchy relations and the semantic relations of classes, we developed artificial nodes/classes to separate siblings classes in different classifiers. These nodes were defined taking in consideration the semantic relations of classes, class-email frequency distribution portrayed in Figure 4.7 (Section 4.2.5), and the structure of the hierarchy portrayed in Figure 4.6 (Section 4.2.5).

Figure 6.4 shows the new hierarchy of classes with artificial classes represented as squares. This class structure display was found empirically and with the support of clustering techniques and semantic analysis and built with the help of the re-structuring methods presented at section 5.2.

First, we transformed the second level of the tree in a two class problem: class 'AMA' was already a class at level 2 which is the parent of a tree-branch that sums up to 30498 emails (64.67% of total



Class	Nr. of Child Classes	Number of Emails	Avg. Nr. Emails / Child Classes
Info_Cidadão (root)	2	47156	23578
AMA	2	30498	15249
Others	23	16658	724,26
AMA/Autenticação.Gov-CC	5	25911	5182,2
AMA/Others	12	4587	199.43

Figure 6.4: Class Structure Re-organization for Classification

data instances) and class ‘Others’, which compile all the other 23 classes at level 2, summing up to 16658 emails (35.33% of total data instances). The next level of class ‘AMA”s branch has 17 categories and a serious class imbalance problem. Five of these classes have a strong semantic relation and where grouped under the artificial class ‘AMA/Autenticação.Gov-CC’, which contains 25911 emails. The remaining classes have no visible semantic relations and where grouped under ‘Others’, which contains 12 classes but only 4587 emails. The class ‘Others’ contains 23 classes which have also an extreme class imbalance problem and since they are all level 2 classes, there is not a strong visible semantic relation between classes to group them.

During this approach we defined a threshold for the exploration of the hierarchy, where models that were classifying classes of level 3 or lower and that could not achieve a performance greater than 80% would not be considered for the final model, to safeguard the accuracy and reliability of our composed model.

To perform classification using this model, the following classification models are needed:

AMA_Others: Classify all data between ‘AMA’ and ‘Other’

1. **AMA/Autenticação.Gov-CC_Others**: Classify ‘AMA’ data between ‘AMA/Autenticação.Gov-CC’ and ‘AMA/Other’
   - (a) **AMA/Autenticação.Gov-CC**: Classify ‘AMA/Autenticação.Gov-CC’ data between all its classes (5)

(b) **AMA/Others**: Classify 'AMA/Others' data between all its classes (12)

2. **Others**: Classify 'Others' data between 23 classes

Model	**Observations**	**Feat. Extraction**	**Feat. Selection**	**Classifier**	**L. maF1**	**O. maF1**	**L. Acc**	**O. Acc**
AMA vs Others		TF-IDF 1-gram	MI 2000 features	XGBoost	0.725	0.725	0.761	0.761
AMA/Autenticação .Gov-CC vs AMA/Others		TF-IDF 1-gram		SGD (OvO - OvR) / XGBoost (tied)	0.912	0.661	0.913	0.695
AMA/Autenticação .Gov-CC	Removed 2 classes that contained less than 0.1% of data	TF-IDF 1-gram		OvR - SGD	0.810	0.535	0.872	0.606
AMA/Others	Removed 4 classes that contained less than 0.1% of data	TF-IDF 1-gram	MI 2000 features	XGBoost	0.788	0.521	0.866	0.602
Others	Removed 7 classes that contained less than 0.1% of data	TF-IDF 1-gram		XGBoost	0.579	0.420	0.631	0.458

L. maF1 = Local Macro-F1; O. maF1 = Overall Macro-F1; L. Acc = Local Accuracy; O. Acc = Overall Accuracy

Table 6.2: Classification Performance for Parent Node-based Approach

Overall Accuracy was calculated using equation 6.4. It is important to notice that with this approach we can predict 64.67% (30498/47156) of data ('AMA') branch to the second level with 76.1% accuracy and third level classes with 60% accuracy.

6.5 Feature-level

As mentioned in section 5.6, there were three major feature selection strategies that where implemented. From all the results observed during the development of this work, we observed that usually using no feature selection method gives a good performance and that the general impact of feature selection methods is small (approximately 1-4%). Chi-squared is the most successful feature selection method, which is in accordance to the literature described in Chapter 3, but there are some situations where TruncatedSVD achieves the best performance.

The performance of the application of the feature selection methods is compared between each other and the use of no feature selection for all ML classifiers applied with TF-IDF for the most tested dataset, Av3_bhc2_M500_mi100_m100, and shown on Figure 6.5.

TF-IDF was present in most of our experiments, and despite TF and Term Presence having had similar performances to TF-IDF in some cases, in most of them they were considerably under TF-IDF.

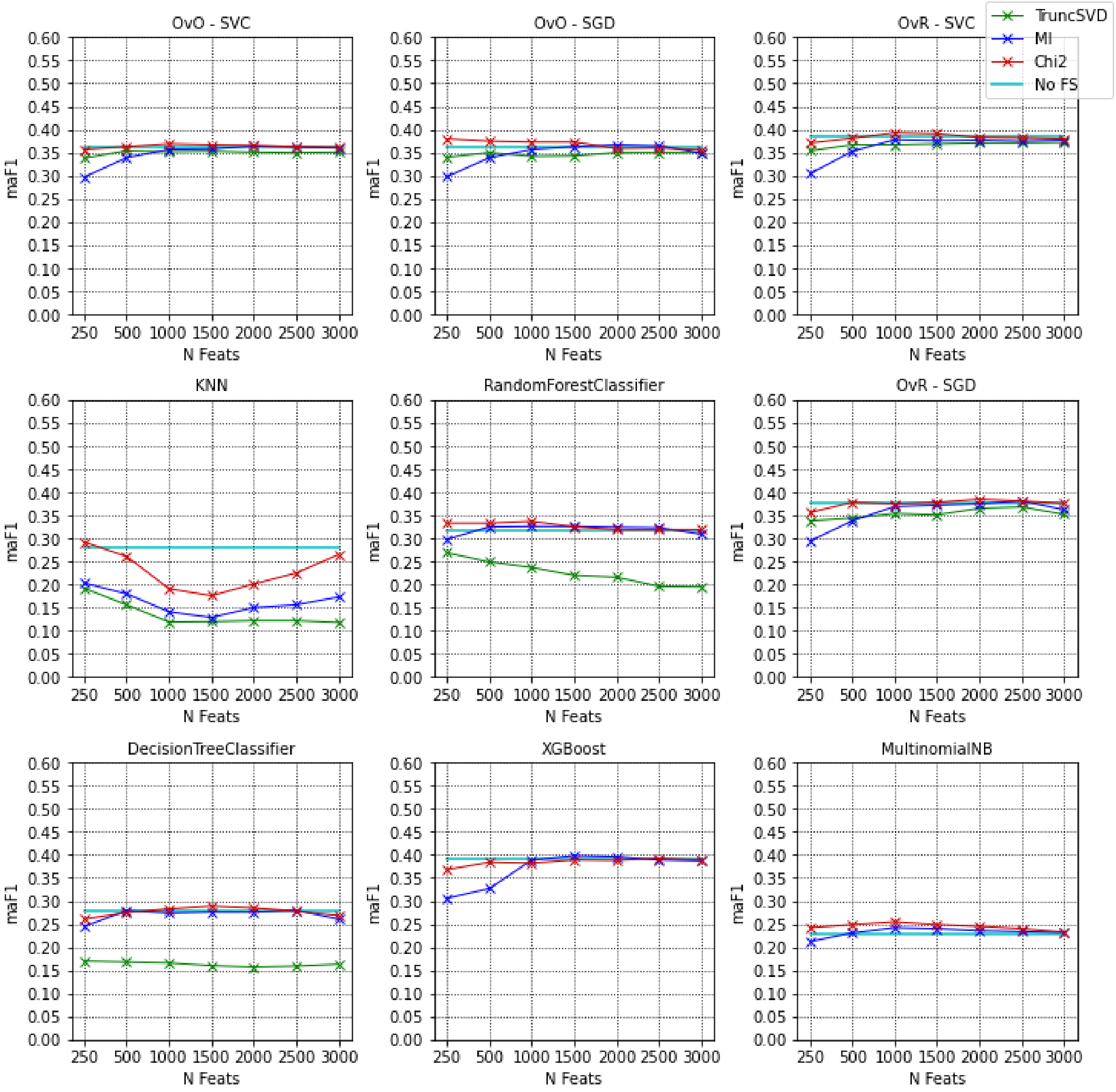


Figure 6.5: Performance of Feature Selection Methods for Av2_bhc2_M500_mi100_m100 - (TF-IDF)

On the other hand, the custom engineered features from text and metadata proved successful, used as standalone feature extraction method and specially when combined with TF-IDF. Table 6.3 shows a small excerpt of the experiments executed with Av3_bhc2_M500_mi100_m100 where we see that we can get 0.301 and 0.19 in two different feature experiments that do not extract any features from text, and that TF-IDF 1-gram increased 0.312% of macro-F1 performance by joining with custom features from text statistics, keywords from all levels of the hierarchy tree, the ‘reply nr’, and the previous category name cut by level 2 and level 4 (complete name of category).

Features Extracted	Feature Selection	macro-F1	Accuracy
TF-IDF + 1-gram + Text Statistics + Keyworks (levels 2 to 4) + Reply nr + Previous Level 2 Category + Previous Level 4 category	Chi^2 1000 features	0.586	0.574
TF-IDF 1-gram	Chi^2 1000 features	0.274	0.271
Keyworks (levels 2 to 4) + Reply nr + Previous Level 2 Category	None	0.301	0.316
Text Statistics + Keyworks level 2 + Reply nr + Previous Level 2 Category	None	0.190	0.203
TF-IDF 1-gram	TruncatedSVD 1500 features	0.562	0.546

Table 6.3: Features Performance for OvO - SVC on Av3_bhc2_M500_mi100_m100

6.6 Classifier-level

Av3_bhc2_M500_mi100_m100 was the most tested dataset because it contains only a fraction of the total quantity of data, turning its train and testing more efficient and contains a small quantity of classes that represent a big proportion of data and do not suffer from much class imbalance.

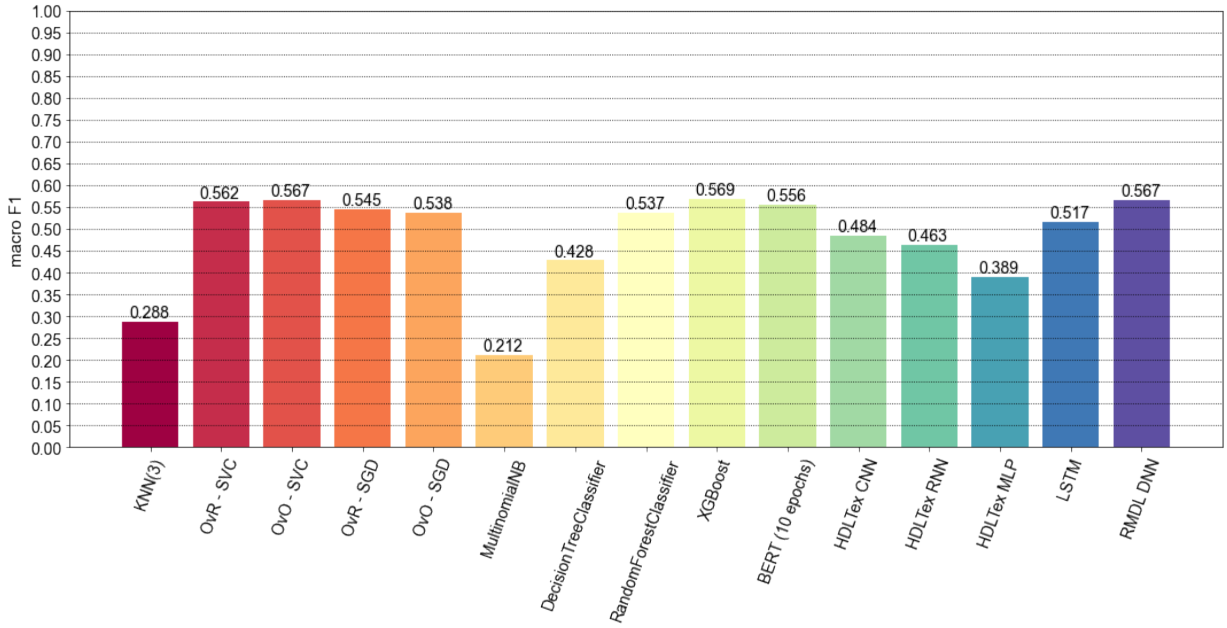


Figure 6.6: Classifiers Performance for Av3_bhc2_M500_mi100_m100 - (ML models trained with TF-IDF and best feature selection method. DL models show the best experimental result)

Figure 6.6 shows a comparison between the different classifiers. In this figure, all the models were trained with TF-IDF weighted features in conjunction with the best performing feature selection method for each classifier. The DL classifiers show the best experimentally attained performance results. Two major observations can be taken from these results:

- Very different algorithms obtain similar performance.
- Several algorithms cannot surpass the 53%-58% performance range.

6.7 Discussion

6.7.1 Flat Classification

The first conclusion we can obtain from the results from flat classification is that substantial improvement can be obtained by improving data quality and by increasing data quantity. We have to take in consideration that each time we increased data quantity we also gained more classes, increasing the complexity of the classification problem and that the proportion of data instances between each class was altered between each iteration.

From the results shown in Figure 6.1 we cannot distinguish the effects on the performance between the increase in data quantity and the improvement in data quality, except in the following situations: from Av2 to Av3_raw, only there was only an increase in data quantity and not in quality and from Av1_raw to Av1 and Av3_raw to Av3 there was an improvement in data quality but not in quantity.

The improvement in data quantity alone shows little improvement in performance, both in macro-F1 measure and in accuracy, but there was an increase in class quantity, so the fact that macro-F1 performance did not drop in face of increasing complexity may be a proof of the positive side of data increase.

The two situations where there was data quality increase alone, are distinct because of the strategies employed for data quality improvement. In the first situation (Av1_raw to Av1), data quality increased by removing mainly systematic noise (removing irrelevant data). In the second situation, the measures applied aimed to improve the meaning of relevant data mainly by normalization and correction of data anomalies. The results show that improving relevant data is much more beneficial than removing irrelevant data.

Figure 6.2 shows the effects of some of our proposed class re-structure strategies. Just as oversampling is, generally, a trade-off between class balance and possibly model overfitting and class overlapping (in multi-class cases), and undersampling is, generally, a trade-off between class balance and data (data with important features may be lost), most of our strategies also act as a trade-off. Following the examples in Figure 6.2, the No Only Child (noc) strategy applied from A to B reduce classes to a more concise hierarchy structure and results in improved macro-F1. From B to C, the No Middle Class (nmc) removes inter-level classes (and respective data (22,8%)) that we consider to have a profound impact on performance from our virtual tree (where only leaf-nodes are target classes) view of the classification problem and Aggregate (agg) strategy aggregates ill represented classes (minority classes with very low number of data instances) in parent classes. The improvement in macro-F1 from these strategies is the advantageous and until this point, we consider that we are only trading class fidelity to the original structure for data sparsity, class overlap and class imbalance reduction. From C to D, Merge and Integrate (mi) and Removing classes with less than 100 data instances (m100) dramatically reduced the number of classes, but only reduced data in 5%, meaning that we are removing non representative classes (which is negative because they are the target of our classification, but not so negative because they represent a minimal fraction of data) in exchange for a huge improvement in macro-F1. From D to E, we undersample all classes to 100 data instances. It is a trade-off between class balance and data loss, which results in a positive increase of macro-F1 performance, but at the cost of a huge loss of data.

Overall, from A to D, macro-F1 performance increased remarkably, but 48.3% macro-F1 it is still low, and despite our data was only reduced by 27.9%, most of it was considered errors (middle-node classes), the total number of classes dropped 87.6% which is a big simplification of our problem that does not pay off the performance improvement to 48.3%.

6.7.2 Local Classification Per Level Approach

As mentioned before, dataset Av3_bhc2_M500_mi100_m100 was the subject of most of our experiences, due not only to the fact that it has a low amount of data and classes, making it practical considering that most classification processes are costly in computational resources and time, but also due to the fact that its classes are the most important and where each class domain is more concise. From level to level, Figure 6.3 shows that performance tends to decrease. This may be explained just because the number of classes increases but also because in lower levels data sparsity, data overlap and label noise play a bigger role. In each of the three cases, the application of the Balanced Hierarchical Cut (bhc) when compared with the simple Hierarchical Cut is much beneficial in performance. The former ensures that undersampling is done without the risk of eliminating all data instances from a former lower level class, aiming for the class that resulted from the cut to completely and fairly represent all of its child classes.

As was the case in the flat classification experiences, applying the Merge and Integrate (mi) and Removing classes with less than 100 data instances (m100) strategies dramatically reduced the number of classes while only eliminating a minimal fraction of data. For example, from Av3_bhc3_M500 to Av3_bhc3_M500_mi100_m100, data instances amount reduces only from 11193 to 10837, which is a reduction of only 3.18% that were distributed over 20 classes (40.82% of previous dataset classes) and results in an increase in macro-F1 performance of 16.4%, resulting in a positive 50.2% score.

6.7.3 Local Classification Per Parent Node-based Approach

In the local classification per parent node-based approach strategy we created two artificial levels with two classes, from which results a binary classification problem and the performance results from it are considerably higher than the ones from the multi-class classification problems observed before.

Following the results in Table 6.4, the ‘AMA vs Other’ model classifies data between a more concise class ‘AMA’ and a very sparse class ‘Others’ and obtains 72.5% and 76.1% of macro-F1 and Accuracy, respectively. The other binary classification model ‘AMA/Autenticação.Gov-CC vs AMA/Others’ distinguishes between more concise classes and obtains macro-F1 and Accuracy scores of 91.2% and 91.3%, respectively. Furthermore, ‘AMA/Autenticaçaao.Gov-CC’ and ‘AMA/Others’ perform multi-class classification on the ‘AMA’ class branch data and obtain very good scores of 81% macro-F1 and 87,2% of accuracy and 78.8% macro-F1 and 86.6% accuracy respectively. On the other hand, ‘Others’ performs multiclass classification on the ‘Others’ branch data, a sparse domain with 6 classes (in which one is an artificial class representing 19 classes with low amount of data) where class overlap, class imbalance and data sparsity is more severe, leads to 57.9% macro-F1 and 63.1% accuracy.

6.7.4 Features and Classifier

The combination techniques applied for feature extraction and feature selection have a big impact on performance. Table 6.3 shows that TF-IDF when applied only with a Chi-squared (1000 features) alone may be surpassed in performance by a conjunction of metadata and engineered features that do not include text and that if combined with a certain conjunction of metadata and engineered features may have its performance more than double. Also, the same feature extraction algorithm may have a certain performance when combined with a given feature selection method (Chi-squared) and the double when combined with other feature selection (TruncatedSVD). Furthermore, Figure 6.5 shows how performance may be so impacted by feature selection and the number of features in one classifier (such as the case of Random Forest classifier) and suffer almost no impact, such as the case of SVC. It also shows how performance varies between classifiers. This information is completed by Figure 6.6 which shows the performance of all classification methods applied, including DL algorithms. DL performed worse than expected, but we also have to acknowledge that due to the complexity and computational cost of these algorithms, they were not tested as extensively as ML and ensemble learning algorithms.

6.7.5 Overview and Summary

In summary, flat classification does not seem the most adequate for this kind of problem. It puts aside the hierarchy relations which are important to structure data and even when applied with class re-structure techniques that sacrifices lots of data and classes to obtain better results, its performance is lacking.

Local Classification approaches are more promising. With per level approaches, classifying data only to hierarchically cut classes at level 2, presents the best achieved results and assigns the most important classes to data. Further classifying data to level 3 or 4 is getting more errors than correctly predicted classes and does not seem feasible to the improvement of the email classification process at the contact center. Per parent node-based approaches obtain the best results although they are more complex since they rely on a complex multi model classification strategy.

To compare the local classification per parent node-based approaches with the local classification per level approaches, its important to compare macro-F1 measures, specially at the most relevant level, level 2. To calculate the average macro-F1 at hierarchy level 2, using per class with local classification per parent node-based, we have to take into account the number of classes described in Figure 6.4, 1 class 'AMA' plus 23 classes 'Others', and the performance results reported in table 6.4, 72.5% macro-F1 for 'AMA' and 42.0% for 'Others' 16 classes (7 classes were removed for before the classification step). This means that the average macro-F1 measure at level 2 is 43.8% (given by Equation 6.5).

$$\frac{0.725 * 1 + 0.420 * 16}{17} = 0.438. \tag{6.5}$$

When compared to level based approach with dataset Av3_bhc2_M500_mi100_m100 (a dataset that also has 15 classes at level 2) which has 57% macro-F1 (see HC2-C1 of table 6.1 and Figure 6.3), the parent node-based approaches falls short with 43.3% macro-F1. However, the parent-node based approach ensures a better performance of 72.5% macro-F1 and 76.1% of accuracy for 64.67% of data which is represented by a single class: 'AMA'.

Despite the hierarchical classification approaches having a significant importance in the performance of our system, Figure 6.6 demonstrates that data quality and label noise are the big limitations of our work. Several algorithms, from traditional ML, to complex DL algorithms and state of the art approaches such as BERT, very different in their mechanics, obtained very similar high-scores (in the range of 54%-59% macro-F1). This evidence suggests that performance is hitting a wall, which we consider to be the lack of data quality and the presence of label noise.

This situation hinders our decision on the several approaches and algorithms taken into account in Chapter 5, to be used to build our final proposal for the email classification system. But, having to make a decision at this point, our proposed email classification system is based on a XGBoost model trained on the Av3_bhc2_M500_mi100_m100 dataset (described in HC2-C1 of Figure 6.3 and Table 6.1), using TF-IDF features. XGBoost not only achieved the best performance with almost all the hierarchical approaches and most offspring datasets, but also proved to be very resilient to the methods used as feature selection (Figure 6.5). Our proposed system is illustrated in Figure 6.7.

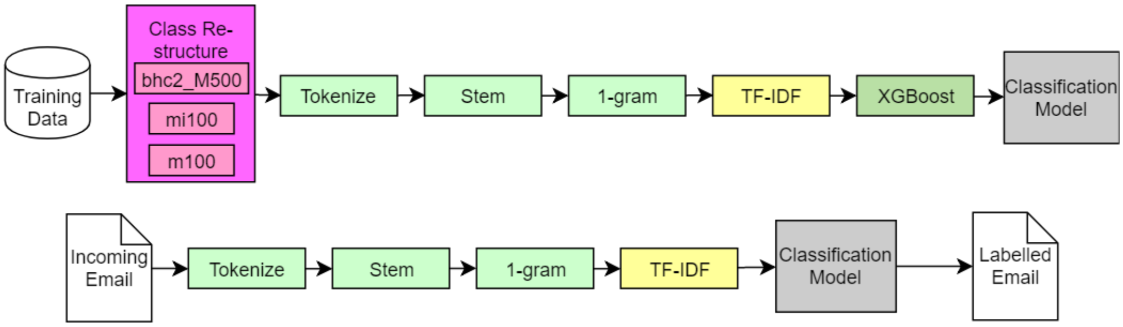


Figure 6.7: Proposed Email Classification System (model training process on top and prediction process on bottom)

7

Conclusion

We don't have better algorithms, we just have more data. More data beats clever algorithms, but better data beats more data

– Peter Norvig, Director of Research, Google

Contents

The main objective, declared in section 1.3, was: to develop an email system able to classify emails of the domain of the Contact Center of the Portuguese Public Services. We have developed a project that evolved a lot from the initial experiences where our best performance was at 30% (macro F1-measure) to the final experiences were our performance was near 60% (macro F1-measure). The proposed email classification system uses the robust and efficient algorithm, XGBoost, and achieves a macro-F1 score of 57%.

Data noise plays a major role in this work. There are few related works in literature based on real-world data. Literature about noisy data describes artificially created noise, not native noise. This project started as a hierarchical classification problem and it was during the development of a solution to that problem that we realized that was not the biggest challenge. At the beginning of iteration 1, we were solving a hierarchical classification problem. At the beginning of iteration 2, we knew that the main problem was not to perform hierarchical classification, but to perform classification with noisy data. By the beginning of iteration 3, after a multitude of strategies regarding noisy data have been applied, we started to suspect another problem even harder to solve: label noise. Label noise corrupts the learning process and greatly impacts the performance of the classification process.

The improvements obtained from the data cleaning and normalization, and the extensive range of algorithms used for classification and their respective performance (where different algorithms achieved similar performance) validate the fact that data noise and label noise are the primary problems that negatively impact the learning and classification processes.

A hierarchy may be very useful for classification, but if it is not properly designed and properly maintained it may pose challenges to automatic classification such as class imbalance, class overlap and label noise. This work proposed new approaches to these challenges, that focus on exploration of class relations and semantic meaning and their consecutive re-structure. The results achieved show that these approaches are successful.

Overall, the work described is this thesis is a testimony of how a problem may unfold into other problems and with that, an extensive study of solutions to the problems of hierarchical classification, noisy data, class imbalance, class overlap and label noise. Meanwhile, we achieved all the challenges listed in the objectives:

1. We successfully collected data from the business context and compiled it into a dataset that iteratively increased in size and quality;
2. We explored the data and where able to identify the many problems that plagued it as well as develop increasingly better solutions to deal with these problems and increase data quality;
3. A compilation of feature selection, feature extraction and dimensionality reduction were not only reviewed but applied and studied with our datasets;
4. The classification strategies applied, range from the most simple and traditional to the complex

deep learning models in recent literature as well as popular state of the art approaches such as BERT.

7.1 Contributions

Section 1.5 summarizes contributions in three separated scopes: data, features and classification. Overall, this thesis gives a comprehensive description of how the combination of the strategies applied in these three scopes can overcome the challenges of the problem of text classification with hierarchical classes, with noisy data and noisy labels.

Our main contribution is a set of strategies that take advantage of the hierarchical relations between classes to counter class issues such as class imbalance. Furthermore, this contribution is accompanied by a report of evolving strategies regarding data cleaning, data normalization, feature extraction, feature selection, dimensionality reduction, feature engineering, text classification using ML, text classification using DL and using novel state of the art approaches, and their performance, to an evolving problem of text classification, that started as a hierarchical classification problem and revealed itself more complex at each step.

Literature about class imbalance asks for new methods to deal with class imbalance, especially in the poorly explored problems of multi-class and hierarchical classification [Esteves, 2020], the few works about label noise state that there are many open research questions to label noise and, for example, methods such as the removal of noisy unreliable data instances should be explored [Frénay and Kabán, 2014]. Class hierarchy maintenance literature underlines the need to explore the hierarchy relationships in favour of its purpose, which is in our case automatic classification [Yuan et al., 2012]. [Kowsari et al., 2017] denotes a research need in identifying new architectures that work with hierarchically structured data.

To the best of our knowledge there are no works in literature about hierarchical classification of real-world data with systematic and natural noise nor works with hierarchy trees similar (in terms of structure and properties) to the one that grounded this problem (described in Figure 4.6).

7.2 Future Work

This thesis is the product of an iterative process of development, shaped by the CRISP-DM methodology. Each iteration saw new approaches to tackle the problems identified in the previous iteration and with each approach new improvements in the performance. However, the iterations that this project saw were limited by data acquisition constraints, time, and human and computational resources. When searching for a solution there are many ways to follow, but limitations such as the aforementioned ones forced us to narrow the solution space, to follow only certain ways leaving aside strategies and methods that were less prioritized and to limit the extent to which the strategies and the methods adopted can be studied and developed.

Future work lays on the continuation of the development process and the overcoming of the limitations aforementioned, which may be summarized as:

- Further improvement of data quality;
- Oversampling techniques and other unexplored class imbalance strategies that were not prioritized during the development of this work;
- Class imbalance techniques specialized on hierarchy text classification is an open field with little to no contributions;
- The use of unsupervised learning strategies to study the class relation and distribution.

Bibliography

[Ali et al., 2019] Ali, H., Salleh, M., Saedudin, R., Hussain, K., and Mushtaq, M. (2019). Imbalance class problems in data mining: A review. *Indonesian Journal of Electrical Engineering and Computer Science*, 14.

[Alizamini et al., 2010] Alizamini, F. G., Pedram, M. M., Alishahi, M., and Badie, K. (2010). Data quality improvement using fuzzy association rules. In *2010 International Conference on Electronics and Information Engineering*, volume 1, pages V1–468. IEEE.

[Arafat et al., 2019] Arafat, M., Hoque, S., Xu, S., and Farid, D. M. (2019). Machine learning for mining imbalanced data. *IAENG International Journal of Computer Science*, 46(2):332–348.

[Atkinson and Metsis, 2020] Atkinson, G. and Metsis, V. (2020). Identifying label noise in time-series datasets. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, pages 238–243.

[Azevedo and Santos, 2008] Azevedo, A. and Santos, M. (2008). Kdd, semma and crisp-dm: A parallel overview. pages 182–185.

[Balakrishnan and Lloyd-Yemoh, 2014] Balakrishnan, V. and Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances.

[Batista et al., 2004] Batista, G., Prati, R., and Monard, M.-C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, 6:20–29.

[Bawden et al., 1999] Bawden, D., Holtham, C., and Courtney, N. (1999). Perspectives on information overload. In *Aslib proceedings*, volume 51, pages 249–255. MCB UP Ltd.

[Bootkrajang, 2016] Bootkrajang, J. (2016). A generalised label noise model for classification in the presence of annotation errors. *Neurocomputing*, 192:61 – 71. Advances in artificial neural networks, machine learning and computational intelligence.

[Brutlag and Meek, 2000] Brutlag, J. D. and Meek, C. (2000). Challenges of the email domain for text classification. In *ICML*, volume 2000, pages 103–110.

[Carvalho and Cohen, 2007] Carvalho, V. R. and Cohen, W. (2007). Recommending recipients in the enron email corpus. *Machine Learning*.

[Carvalho and Cohen, 2005] Carvalho, V. R. and Cohen, W. W. (2005). On the collective classification of email speech acts. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352. ACM.

[Cassier et al., 2019] Cassier, M., Sellami, Z., and Lorré, J.-P. (2019). Meeting Intents Detection Based on Ontology for Automatic Email Answering. In *Journées francophones d'Ingénierie des Connaissances (IC 2019)*, Toulouse, France.

[Chi et al., 2019] Chi, S., Qiu, X., Xu, Y., and Huang, X. (2019). How to fine-tune bert for text classification?

[Clark et al., 2003] Clark, J., Koprinska, I., and Poon, J. (2003). A neural network based approach to automated e-mail classification. pages 702 – 705.

[Cohen, 1996] Cohen, W. W. (1996). Learning rules that classify e-mail. In *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning and Information Access*.

[Cruz et al., 2019] Cruz, R. M., Souza, M. A., Sabourin, R., and Cavalcanti, G. D. (2019). Dynamic ensemble selection and data preprocessing for multi-class imbalance learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(11):1940009.

[Denning, 1982] Denning, P. J. (1982). Acm president's letter: Electronic junk. *Commun. ACM*, 25(3):163–165.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dou et al., 2015] Dou, D., Wang, H., and Liu, H. (2015). Semantic data mining: A survey of ontology-based approaches. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pages 244–251.

[Dredze et al., 2009] Dredze, M., Schilit, B. N., and Norvig, P. (2009). Suggesting email view filters for triage and search. In *Twenty-First International Joint Conference on Artificial Intelligence*.

[Esteves, 2020] Esteves, V. M. S. (2020). Techniques to deal with imbalanced data in multi-class problems: A review of existing methods.

[Favaretto et al., 2019] Favaretto, M., De Clercq, E., and Elger, B. S. (2019). Big data and discrimination: perils, promises and solutions. a systematic review. *Journal of Big Data*, 6(1):12.

[Flores and Moreira, 2016] Flores, F. N. and Moreira, V. P. (2016). Assessing the impact of stemming accuracy on information retrieval – a multilingual perspective. *Information Processing Management*, 52(5):840 – 854.

[Frénay and Kabán, 2014] Frénay, B. and Kabán, A. (2014). A comprehensive introduction to label noise. In *ESANN*.

[Frénay and Verleysen, 2013] Frénay, B. and Verleysen, M. (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[Galar et al., 2011] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484.

[Gutiérrez et al., 2016] Gutiérrez, P. A., Pérez-Ortiz, M., Sánchez-Monedero, J., Fernández-Navarro, F., and Hervás-Martínez, C. (2016). Ordinal regression methods: Survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146.

[Haixiang et al., 2017] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., and Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220 – 239.

[Heidarysafa et al., 2018] Heidarysafa, M., Kowsari, K., Brown, D. E., Meimandi, K. J., and Barnes, L. E. (2018). An improvement of data classification using random multimodel deep learning (rmdl). *arXiv preprint arXiv:1808.08121*.

[Henderson et al., 2017] Henderson, M., Al-Rfou, R., Strope, B., Sung, Y.-h., Lukács, L., Guo, R., Kumar, S., Miklos, B., and Kurzweil, R. (2017). Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

[Honnibal and Montani, 2017] Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

[Horvitz et al., 1999] Horvitz, E., Jacobs, A., and Hovel, D. (1999). Attention-sensitive alerting. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 305–313. Morgan Kaufmann Publishers Inc.

[Japkowicz, 2000] Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*, volume 56. Citeseer.

[John, 1962] John, A. (1962). How to do things with words.

[Kannan et al., 2016] Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A., Miklos, B., Corrado, G., Lukács, L., Ganea, M., Young, P., and Ramavajjala, V. (2016). Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 955–964.

[Katakis et al., 2007] Katakis, I., Tsoumakas, G., and Vlahavas, I. (2007). E-mail mining: Emerging techniques for e-mail management. In *Web Data Management Practices: Emerging Techniques and Technologies*, pages 220–243. IGI Global.

[Ke et al., 2006] Ke, S.-W., Bowerman, C., and Oakes, M. (2006). Perc: A personal email classifier. In *European Conference on Information Retrieval*, pages 460–463. Springer.

[Kingma and Ba, 2017] Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

[Klimt and Yang, 2004] Klimt, B. and Yang, Y. (2004). The enron corpus: A new dataset for email classification research. volume 3201, pages 217–226.

[Kowsari et al., 2019] Kowsari, Meimandi, J., Heidarysafa, Mendu, Barnes, and Brown (2019). Text classification algorithms: A survey. *Information*, 10(4):150.

[Kowsari et al., 2017] Kowsari, K., Brown, D. E., Heidarysafa, M., Jafari Meimandi, K., Gerber, M. S., and Barnes, L. E. (2017). Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371.

[Kowsari et al., 2018] Kowsari, K., Heidarysafa, M., Brown, D. E., Meimandi, K. J., and Barnes, L. E. (2018). Rmdl. *Proceedings of the 2nd International Conference on Information System and Data Mining - ICISDM '18*.

[Labadie and Prince, 2008] Labadie, A. and Prince, V. (2008). The impact of corpus quality and type on topic based text segmentation evaluation. In *2008 International Multiconference on Computer Science and Information Technology*, pages 313–319.

[Lavanya et al., 2014] Lavanya, S., Palaniswami, D. S., and Sudha, S. (2014). Efficient methods to solve class imbalance and class overlap.

[Li et al., 2017] Li, Y., Yang, J., Song, Y., Cao, L., Luo, J., and Li, J. (2017). Learning from noisy labels with distillation. *CoRR*, abs/1703.02391.

[Liu et al., 2020] Liu, Q., Kusner, M. J., and Blunsom, P. (2020). A survey on contextual embeddings.

[Manning et al., 2008] Manning, C. D., Schütze, H., and Raghavan, P. (2008). *Introduction to information retrieval*. Cambridge university press.

[Meadow et al., 2000] Meadow, C., Boyce, B., and Kraft, D. (2000). *Text Information Retrieval Systems*. Library and information science. Academic Press.

[Mitra et al., 2014] Mitra, N., Goel, N., Chakraverty, S., and Singh, G. (2014). Adaptive content based textual information source prioritization. *ICTACT Journal on Soft Computing*, 5(1).

[Moreira and Huyck, 2001] Moreira, V. and Huyck, C. (2001). A stemming algorithmm for the portuguese language. pages 186– 193.

[Mujtaba et al., 2017] Mujtaba, G., Shuib, L., Raj, R. G., Majeed, N., and Al-Garadi, M. A. (2017). Email classification research trends: Review and open issues. *IEEE Access*, 5:9044–9064.

[Müller and Markert, 2019] Müller, N. M. and Markert, K. (2019). Identifying mislabeled instances in classification datasets. *CoRR*, abs/1912.05283.

[Neustaedter et al., 2005a] Neustaedter, C., Brush, A., and Smith, M. A. (2005a). Beyond from and received: Exploring the dynamics of email triage. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1977–1980. ACM.

[Neustaedter et al., 2005b] Neustaedter, C., Brush, A. B., Smith, M. A., and Fisher, D. (2005b). The social network and relationship finder: Social sorting for email triage. In *CEAS*.

[Nicolosi, 2008] Nicolosi, N. (2008). Feature selection methods for text classification. *Department of Computer Science, Rochester Institute of Technology, Tech. Rep*.

[Park et al., 2019] Park, S., Zhang, A. X., Murray, L. S., and Karger, D. R. (2019). Opportunities for automating email processing: A need-finding study. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 374. ACM.

[Piatetsky, 2014] Piatetsky, G. (2014). Crisp-dm, still the top methodology for analytics, data mining, or data science projects. *KDD News*.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14:130–137.

[Roberts et al., 2010] Roberts, P. J., Howroyd, J., Mitchell, R., and Ruiz, V. (2010). Identifying problematic classes in text classification. In *2010 IEEE 9th International Conference on Cyberntic Intelligent Systems*, pages 1–6.

[Salih et al., 2019] Salih, F. I., Ismail, S. A., Hamed, M. M., Yusop, O. M., Azmi, A., and Azmi, N. F. M. (2019). Data quality issues in big data: A review. In *Recent Trends in Data Science and Soft Computing*, pages 105–116. Springer.

[Sarrafzadeh et al., 2019a] Sarrafzadeh, B., Awadallah, A. H., and Shokouhi, M. (2019a). Exploring email triage: Challenges and opportunities. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, pages 325–329. ACM.

[Sarrafzadeh et al., 2019b] Sarrafzadeh, B., Hassan Awadallah, A., Lin, C. H., Lee, C.-J., Shokouhi, M., and Dumais, S. T. (2019b). Characterizing and predicting email deferral behavior. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 627–635. ACM.

[Schafer and Graham, 2002] Schafer, J. L. and Graham, J. W. (2002). Missing data: our view of the state of the art. *Psychological methods*, 7(2):147.

[Searle and Searle, 1969] Searle, J. R. and Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

[Shanmugalingam et al., 2019] Shanmugalingam, K., Chandrasekara, N., Hindle, C., Fernando, G., and Gunawardhana, C. (2019). Corporate it-support help-desk process hybrid-automation solution with machine learning approach. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7.

[Silla and Freitas, 2011] Silla, Jr., C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72.

[Silva et al., 2018] Silva, S., Ribeiro, R., and Pereira, R. (2018). Less is more in incident categorization.

[Sinha et al., 2018] Sinha, K., Dong, Y., Cheung, J. C. K., and Ruths, D. (2018). A hierarchical neural attention-based text classifier. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 817–823, Brussels, Belgium. Association for Computational Linguistics.

[Snelting and Tip, 1998] Snelting, G. and Tip, F. (1998). Reengineering class hierarchies using concept analysis. In *Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '98/FSE-6, page 99–110, New York, NY, USA. Association for Computing Machinery.

[Souza et al., 2020] Souza, F., Nogueira, R., and Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*.

[Spark Jones, 1972] Spark Jones, K. (1972). A statistical interpretation of term importance in automatic indexing. *Journal of Documentation*, 28(1):11–21.

[Tan, 2005] Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671. Cited By :207.

[Tang et al., 2006] Tang, L., Zhang, J., and Liu, H. (2006). Acclimatizing taxonomic semantics for hierarchical content classification. volume 2006, pages 384–393.

[The Radicati Group, 2015] The Radicati Group, I. (2015). Email statistics report, 2015-2019. Technical report.

[Toman et al., 2006] Toman, M., Tesar, R., and Jezek, K. (2006). Influence of word normalization on text classification.

[Van der Aalst et al., 2018] Van der Aalst, W. M., Bichler, M., and Heinzl, A. (2018). Robotic process automation.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

[Vijayarani et al., 2015] Vijayarani, S., Ilamathi, M. J., and Nithya, M. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16.

[Whittaker and Sidner, 1997] Whittaker, S. and Sidner, C. (1997). Email overload: exploring personal information management of email. *Culture of the Internet*, pages 277–295.

[Wibowo Haryanto et al., 2018] Wibowo Haryanto, A., Kholid Mawardi, E., and Muljono (2018). Influence of word normalization and chi-squared feature selection on support vector machine (svm) text classification. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 229–233.

[Wille, 1982] Wille, R. (1982). Restructuring lattice theory: An approach based on hierarchies of concepts. In Rival, I., editor, *Ordered Sets*, pages 445–470, Dordrecht. Springer Netherlands.

[Williams et al., 2015] Williams, J. D., Kamal, E., Ashour, M., Amr, H., Miller, J., and Zweig, G. (2015). Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis). In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 159–161.

[Xia et al., 2007] Xia, Y., Wang, J., Zheng, F., and Liu, Y. (2007). A binarization approach to email categorization using binary decision tree. In *2007 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3459–3464.

[Xiong et al., 2010] Xiong, H., Wu, J., and Liu, L. (2010). Classification with classoverlapping: A systematic study. pages 303–309. Atlantis Press.

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

[Yoo et al., 2011] Yoo, S., Yang, Y., and Carbonell, J. (2011). Modeling personalized email prioritization: classification-based and regression-based approaches. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 729–738. ACM.

[Yu and hua Zhu, 2009] Yu, B. and hua Zhu, D. (2009). Combining neural networks and semantic feature space for email classification. *Knowledge-Based Systems*, 22(5):376 – 381.

[Yuan et al., 2008] Yuan, P., Chen, Y., Jin, H., and Huang, L. (2008). Msvm-knn: Combining svm and k-nn for multi-class text classification. In *IEEE International Workshop on Semantic Computing and Systems*, pages 133–140.

[Yuan et al., 2012] Yuan, Q., Cong, G., Sun, A., Lin, C.-Y., and Thalmann, N. (2012). Category hierarchy maintenance: A data-driven approach.

[Zhu and Wu, 2004] Zhu, X. and Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210.

[Zhuge and He, 2017] Zhuge, H. and He, L. (2017). Automatic maintenance of category hierarchy. *Future Generation Computer Systems*, 67:1 – 12.

A

Email Overload

A.1 Email Overload

Since email overload has been mentioned in 1982 [Denning, 1982, Whittaker and Sidner, 1997, Bawden et al., 1999], email management importance has been increasing. As the volume of emails grow, the user's time spent processing emails increase. It has been estimated that emails, in the context of work, receive an average of 100 emails per day [Carvalho and Cohen, 2007, The Radicati Group, 2015]. It has been estimated that users, in the context of work, receive an average of 100 emails per day [The Radicati Group, 2015, Carvalho and Cohen, 2007].

[Katakis et al., 2007] explore how Machine Learning (ML) and Data Mining (DM) can contribute as a solution to the email overload problem [Denning, 1982, Whittaker and Sidner, 1997, Bawden et al., 1999], stating that "email categorization into folders, email answering and summarization, spam filtering, are only a few representatives".

A.1.1 Email Management and Processing Research

Generally, the main tool for email management is email classification [Cohen, 1996, Brutlag and Meek, 2000], where emails are classified into one or more of a discrete set of predefined categories. Currently, popular email clients allow users to classify emails into important or normal, starred or normal, spam or not spam categories, using machine learning techniques [Yoo et al., 2011, Ke et al., 2006].

However, email overload suggests that these binary models are not enough to manage email attention. [Park et al., 2019] presented a need-finding study to identify the needs of the email users, regarding the automation of their email experience. The study conducted three different experiments. The first experiment conducted a workshop to gather ideas from users ”for automating email in their own inboxes” and found out that 47% of the ideas could not be expressed by common email clients (for instance Gmail and Oultook). The second experiment studied how email users where already “automating aspects of their email” (by mining public repositories on GitHub that make use of IMAP) to see which needs have been addressed and found that 90% of those automations didn't exist in common email clients. They examined automations implemented by programmers, who, having the ability to do it, carried out their desires for personalized automation. In the third experiment,a platform called YouPS was developed to allow users to write, test, debug, save and execute simple Python scripts that manipulate emails arriving in inbox. With a platform, that allows users to “easily” test and run their scripts over their own email, they invited 12 subjects to write and execute scripts for a week and then examined their scripts and interviewed them to find out about their experiences, and therefore their needs. The results where that 40% of their needs were not possible to be fulfilled by common email clients.

The needs identified in these studies were grouped in the following categories:

1. Triage and prioritization (of incomming emails);
2. Inbox folder management (more interactions with the inbox folder: move emails to different locations, shared inboxes, sort inbox emails);
3. Categorization of emails (create and tag emails with labels);
4. Email sending automatization (easily/automatically draft responses and reply);
5. Email “modes” (toggle email modes, like ”vacation”, ”work”, ”sleep”, ...);
6. Inbox presentation (alternative views for email threading, different views for email messages);
7. Aggregation and processing of multiple emails (aggregate multiple emails in a response, group multiple emails information in a summary, chart or visualization model).

These seven categories of email automation needs/opportunities are a reflection of the problem of email overload. The identification of these challenges is not recent. Most of them have been tackled since they were identified, especially spam filtering which, due to its financial impact, has been observed to gather most of the attention [Katakis et al., 2007, Mujtaba et al., 2017].

Mujtaba et al., 2017 reviews articles on e-mail classification published between 2006 and 2016 and confirms this trend, showing that almost two thirds (63 in 98 analyzed) of the articles are related to spam and/or phishing email classification. Multi-folder Categorization in which researchers develop a multi-class classifier that categorizes emails into various predefined categories (for example user-defined directories) comes second with 20/98 articles and is in line with the scope of this project. Also “Interesting or Uninteresting Email Classification”, “Inquiry Email Classification”, “Complaint Email Classification”, “Private or Official Email Classification” all with only 1 in 98 articles examined meet the needs of email attention management [Park et al., 2019].

A.1.2 Triage and Prioritization

The first process identified in the email processing process in figure 1.1 was Email Triage.

The Oxford Dictionary defines triage as ”(in a hospital) the process of deciding how seriously ill/sick or injured a person is, so that the most serious cases can be treated first” and prioritization as ”the act of putting tasks, problems, etc. in order of importance, so that you can deal with the most important first”. Similarly, email triage is defined as ”the process of going through unhandled email and deciding what to do with it” [Neustaedter et al., 2005a, Neustaedter et al., 2005b, Sarrafzadeh et al., 2019a, Sarrafzadeh et al., 2019b] or simply as ”sorting email” [Dredze et al., 2009].

Prioritization of email is understood as ranking emails by discrete levels of importance, based on their relevance/importance [Mitra et al., 2014]. Email triage research papers usually refer Prioritization as a sub-process of Triage, namely the stage where users decide what emails to handle first (by priority vs sequentially) [Neustaedter et al., 2005a, Sarrafzadeh et al., 2019a, Sarrafzadeh et al., 2019b].

A.1.3 Email Categorization

From the opportunities/needs identified in section A.1.1 to deal with the problem of email overload, Inbox Folder Management, Categorization of email, Email sending automatization and Triage and prioritization, all count with email classification strategies.

Inbox folder Management and Categorization of emails suggest that emails should be divided into folders or classified with labels to deal with email overload.

Email sending automatization, proposes to automatically send email replies. To send email replies, emails first need to be categorized and then a reply needs to be generated. The answer may be automatically generated [Kannan et al., 2016, Henderson et al., 2017, Cassier et al., 2019] or there may be a set of predefined answers that are related with the category assigned to the email (the contact center case).

Triage and prioritization, namely prioritization, is a subject that also involves email categorization, because prioritization is the process of assigning an urgency or importance level to the email, and therefore makes use of email classification techniques to classify emails in a Likert scale of urgency or importance.

So, it is safe to say that the main subject of this research is email categorization. Email categorization, email classification or email category prediction is the problem of linking an email with a category label. In the email context, a category is usually a label that represents the intent, the summary subject, the action required, the topic, etc, characterized/described in the email.

To understand how email categorization may be done, first we need to understand what is an email.

A.1.4 Emails as Data

The type of data that we have in this work are emails.

An email is a semi-structured source of information [Katakis et al., 2007]. Structured, because it follows a structure: *to*, *from*, *cc*, *bcc*, *date*, *subject* and *body*. Semi because the body, which is the largest source of information, is written in natural language.

These properties may be grouped under header features (”to”, ”from”, ”bcc”, ”cc” and ”subject”) and body features (”body” and ”attachments”). Furthermore, since each email is sent at a time point and may start or be part of a chain of emails, we may consider the time of sending, the relation with the previous and the next emails and its position in the chain of emails as a metadata.

The body and the subject of the email usually contain the features that best discriminate the email. This means that the main source of information from our data is text, and the problem of email classification can ultimately be reduced to a problem of text classification, a well known and popular research domain.

A.2 Email Triage Background

Both Carman Neustaedter et al. [Neustaedter et al., 2005a, Neustaedter et al., 2005b] and Bahareh Sarrafzadeh et al. [Sarrafzadeh et al., 2019a, Sarrafzadeh et al., 2019b] investigate the problem of email triage by formalizing interview and survey results about user's triage behaviours into rules and processes, giving a more concise comprehension of the subject. Both see Triage and Prioritization as an approach to deal with email overload [Denning, 1982, Bawden et al., 1999, Whittaker and Sidner, 1997].

Neustaedter et al. [Neustaedter et al., 2005a] interviewed ten information workers and then distributed a survey to 2000 employees getting 233 responses. Sarrafzadeh et al. [Sarrafzadeh et al., 2019b] also employed a study methodology of interview and survey. They conducted an interview with 15 information workers and second and distributed a survey to different mailing lists within that company, having collected 91 responses.

Users can either use a single pass or multiple passes to when they triage their emails, which means handling all mails that come across starting at the top or bottom of the list or handling only certain types of emails in each pass, respectively. Multiple pass users usually scan the same email more than once before handling it.

Users reported on how they handle emails regarding order (sequentially or by priority). Neustaedter et al. [Neustaedter et al., 2005a] stated that all single-pass users handle emails sequentially while multiple-pass users handle a certain type of emails in each pass, either sequentially or by priority. Fig. A.1 shows the results from both Neustaedter et al. [Neustaedter et al., 2005a] and Sarrafazadeh et al. [Sarrafzadeh et al., 2019b]'s studies regarding the order in which the users handle their emails and the number of passes (multiplicity).

Multiplicity	Single-Pass	Multiple-pass	Both	Neutral
Neustaedter	17%	47%	9%	27%
Sarrafzadeh	46%	51-54%		
Order	Sequentially	Priority	Both	Neutral
Neustaedter	30%	19%	15%	36%
Sarrafzadeh	48%	41%		

Table A.1: Results regarding triage strategies [Sarrafzadeh et al., 2019b, Neustaedter et al., 2005a]

Carman Neustaedter et al. [Neustaedter et al., 2005a] described Triage as "Look, Act", where "Look" means literally looking, without opening the email and "Act" means to read, not read, move, leave, reply, delete or performing another task.

Sarrafzadeh et al. [Sarrafzadeh et al., 2019b] defined triage is going through a list of emails and making a decision over each one. Deciding how to handle an email comprehends three steps: Prioritization, Deferral, Strategy to Facilitate Revisiting.

1. Prioritization: the user may use different strategies to measure the level of attention that each email requires.

2. Deferral: the user needs to decide whether to deal with email immediately or later. Several factor impact the decision of whether the message is deferred including: time or effort needed, urgency, sender and workload.

3. Strategies to Facilitate Revisiting: When the email is deferred, the user has to deal with it in the future. Users may take actions to facilitate dealing with deferred emails. Two types of email

management strategies were identified:

(a) Preparatory Organization encompasses managing attention and anticipating contact retrieval with approaches ranging from flagging or marking emails as unread to creating tasks and ”ToDo” lists in planners and calendars.

(b) Opportunistic Management leaves context retrieval to the time of retrieval, which basicaly leads to re-triage.

Both strategies are costly and impact productivity.

Email Prioritization focuses on making a personalized prediction of the importance label of non-spam emails.

Starting as early as 1999, Horvitz, Jacobs, and Hovel [Horvitz et al., 1999], took the perspective that human attention is the most valuable and scarcest commodity in human-computer and interaction, dwelling on the challenge of attention management, explore the problem of inferring the expected ”criticality” of email. The family of systems/prototypes developed in this work (called ”PRIORITIES”) use a SVM classifier coupled with importance-specific tokens to classify emails into importance classes. These tokens include labels and patterns such as the Sender, Recipients, Time Criticality (inferred time of an implied meeting, language indicating cost with delay), Past tense, Future-tense, Future dates, Coordination (language used to refer to coordinative tasks), Personal requests, Importance (language referring to importance), Length of message and Presence of attachments.

Neustaedter et al. [Neustaedter et al., 2005b] present the term ”social sorting” which is the re-ranking of collections of email based on social attributes. It defines metrics for measuring the social importance of users based on the email elements (from, to, cc) and user actions of reading or replying, which can potentially be used for measuring email prioritization.

Triage is an intrinsic email management process. We presented two views of defining the triage. Simply as ”Look, Act” or as ”going through a list of emails and making a decision over each one”, where the decision comprehends three steps (prioritization, deferral and strategies to facilitate revisiting). It may be characterized according to the order how emails are handled (sequentially/by priority) and to the multiplicity of the process (single-pass/multiple pass).

B

Data Description

B.1 Raw Data

Data collected from the API was store in multiple JSON files and there was no documentation. All information about the data collected had to be inferred. Each JSON file represents all the actions following the first email received, regarding some issue, until the issue (”ticket” in the business context) is closed. The JSON schema was inferred and is presented in Listings B.1

Listing B.1: Inferred schema from json files

```
{
    ”$schema”: ”http://json-schema.org/draft-04/schema#”,
    ”type”: ”object”,
    ”properties”: {
        ”domain”: { ”type”: ”string”},
        ”email_quality_log”: {
            ”type”: ”array”,
            ”items”: { ”items”: {}, ”additionalItems”: true},
            ”additionalItems”: true
        },
        ”episodes”: {
            ”type”: ”array”,
            ”items”: {
                ”type”: ”object”,
                ”properties”: {
                    ”attachments”: {
                        ”type”: ”array”,
                        ”items”: { ”items”: {}, ”additionalItems”: true },
                        ”additionalItems”: true },
                    ”close_comments”: { ”type”: [ ”string”, ”null” ] },
                    ”description”: { ”type”: ”null” },
                    ”episode_create_date”: { ”type”: ”string” },
                    ”episode_subject”: { ”type”: [ ”string”, ”null” ] },
                    ”html_part”: { ”type”: [ ”string”, ”null” ] },
                    ”queue_name”: { ”type”: ”string”,”enum”: [”Info Cidad[U+FFFD]o”] },
                    ”queue_uuid”: { ”type”: ”string” },
                    ”sent_to”: { ”type”: [ ”string”, ”null” ] },
                    ”status”: { ”type”: ”string”,”enum”: [ ”CLOSED”, ”OPEN”, ”NEW”, ”FW_CLOSED”,
                         ”FW_WAIT”
                        ]},
                    ”subject_uuid”: { ”type”: [ ”string”, ”null”  ] },
                    ”text_part”: { ”type”: [ ”null”, ”string” ] },
                    ”ticket_tags”: {”type”: ”null” },
                    ”type”: { ”type”: ”string”, ”enum”: [”OUTBOUND”,”SETAGENT”,”INBOUND”,”OPEN”
                         ] },
                    ”user_name”: {”type”: [ ”string”, ”null” ]
                    },
                    ”uuid”: { ”type”: ”string” },
                    ”visibility”: { ”type”: ”string”, ”enum”: [”PUBLIC”, ”PRIVATE” ] }
                }
            },
            ”additionalItems”: true
        },
        ”occurrence”: { ”type”: ”integer” },
        ”queue_name”: { ”type”: ”string” },
        ”queue_uuid”: { ”type”: ”string” },
        ”scripts_info”: {
            ”type”: ”array”,
            ”items”: {
                ”items”: {},
                ”additionalItems”: true },
            ”additionalItems”: true },
        ”source_type”: { ”type”: ”string” },
        ”status”: { ”type”: ”string” },
        ”subject”: { ”type”: ”string” },
        ”ticket_create_date”: { ”type”: ”string” },
        ”ticket_expires_on”: { ”type”: ”string” },
        ”ticket_id”: { ”type”: ”string”  },
        ”user_id”: { ”type”: ”integer” }
    }
}
```

B.2 Common Offspring Datasets

Dataset	Emails/Class								
	Emails	Classes	Min	Max	Mean	Q1	Median	Q3	IQR
Av3	49564	138	1	5676	359.16	6	51.5	297.5	291.5
Av3_hc2	49564	25	2	30498	1982.56	25	252	1569	1544
Av3_hc3	49564	62	1	22002	799.42	6.25	97	444	437.75
Av3_hc4	49564	126	1	5676	393.37	6	54	311.5	305.5
Av3_hc5	49564	138	1	5676	359.16	6	51.5	297.5	291.5
Av3_noc	49564	129	1	5676	384.22	7	72	313	306
Av3_noc_ag100a	49564	108	1	5676	458.93	8.75	127.5	402.75	394
Av3_noc_ag100a_nmc	44347	91	1	5676	487.33	9	140	406.5	397.5
Av3_noc_ag100a_nmc_mi100_m100	43802	54	104	5676	811.15	200.75	328.0	605.5	404.75
Av3_noc_ag100a_nmc_mi100_m100_u100	5616	54	104	104	104	104	104	104	0
Av3_nmc	42993	112	1	5676	383.87	6	63	308.5	302.5
Av3_noc_nmc	44419	111	1	5676	400.17	7	88	315.5	308.5
Av3_noc_nmc_m100	43319	53	100	5676	817.34	200	327	607	407
Av3_bhc2_M500	6259	24	2	500	260.79	20.75	249.5	500	479.25
Av3_bhc2_M500_m100	6073	15	129	500	404.87	304	500	500	196
Av3_bhc2_M500_mi100	6259	24	2	500	260.79	20.75	249.5	500	479.25
Av3_bhc2_M500_mi100_m100	6073	15	129	500	404.87	304	500	500	196
Av_bhc3_M500	11193	49	1	500	228.43	9	136	500	491
Av3_bhc3_M500_m100	10837	29	104	500	373.69	247	454	500	253
Av3_bhc3_M500_mi100	11193	44	2	500	254.39	45.25	242	500	454.75
Av3_bhc3_M500_mi100_m100	10837	29	104	500	373.69	247	454	500	253
Av3_bhc4_M500	17788	104	1	500	171.04	6	78.5	320.25	314.25
Av3_bhc4_M500_m100	16674	48	100	500	347.38	192.25	374.5	500	307.75
Av3_bhc4_M500_mi100	17788	77	1	500	231.01	46	172	500	454
Av3_bhc4_M500_mi100_m100	17149	51	100	500	336.25	175.5	327	500	324.5

Table B.1: Common Offspring Datasets Statistics

C

Algorithms

C.1 Text Representation

The most usual are:

One-hot encoding: is one of the simplest forms of representation, one-hot encodings convert a single word into a vector of N dimensions, filled with zeros and with a single (hot) position with a one. N is the size of the dictionary being used, where each position represents a word. It is computationally cheap to convert data to one hot encoding data and each vector carries only bare information.

BOW: is a representation where text is converted into a vector, vocabulary sized, where each index represent a word and its value represents the word frequency in the text. This model only represents the words and their frequency. It does not represent the order of the words neither their meaning.

WE: is a mapping of words into vectors, where each word gets converted into a vector of N dimensions. Each dimension has a meaning and the values of these dimensions represent the meaning of the word. Different words with similar meanings will have similar vectors. There are generally two approaches for using a word embedding in a problem: the WE model may be trained from the data being used in the problem or a pre-trained WE model may be used. Training a model is an computationally expensive process, so for most problems, the best approach is using a pre-trained WE model. Whoever, some corpus have very specific vocabulary, where training a WE model would be a good approach to represent the data. Despite being more sophisticated than the previous representations fall short when dealing with multiple meanings, misspellings, neologisms and OOVs.There are four popular approaches to calculate word embeddings: Word2Vec, GloVe, FastTexc and using a Word Embedding Layer.

Contextual Embedding: "assigns each word a representation based on its context, thereby capturing uses of words across varied contexts and encoding knowledge that transfers across languages" [Liu et al., 2020].

C.2 Feature Extraction Algorithms

Term Frequency (TF) is the most basic form of weighted terms feature extraction, where each term is mapped to a number corresponding to the number of occurrences of that word in a document. It is given by equation:

$$\mathrm{TF_{d,t}} = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document } d} \tag{C.1}$$

Term Presence (TP) is basically a boolean TF where each term is mapped to 1 if it occurs at least one time in a given document or 0 otherwise.

Inverse Document Frequency (IDF) is a measure where each term is mapped to the quotient between the number of documents in a collection/corpus and the number of documents where the terms occurs at least one time in the same collection/corpus, given by equation:

$$\mathrm{IDF_t} = \frac{\text{Total number of documents}}{\text{Number of documents where term } t \text{ appear}} \tag{C.2}$$

Term Frequency - Inverse Document Frequency (TF-IDF) is a popular statistic proposed by [Spark Jones, 1972] which reflects how important a term is to a document in a collection/corpus. It is the combination of TF and IDF and is given by equation:

$$\mathrm{TF\text{-}IDF_{d,t}} = \frac{TF_{d,t}}{IDF_t} \tag{C.3}$$

C.3 Feature Selection Algorithms

Chi Squared is used in statistics to test the independence of two variables (in this work context: terms). Given the data of two variables, we can get observed count O and expected count E. Chi-Square measures how expected count E and observed count O deviates each other. If Chi^2 test value is high it means that there is a correlation between two variables where the opposite means that there is no correlation. This feature selection methods ranks terms by Chi^2 test value and selects the n highest terms.

Mutual Information measures the reduction in uncertainty for one variable given a known value of the other variable. measures the average reduction in uncertainty about one variable that results from learning the value of another value; or vice versa, the average amount of information that one variable conveys about other variable. This measure is used to rank features and select the top n.

Information Gain (IG) is the same as Mutual Information [Nicolosi, 2008, Manning et al., 2008].

Truncated Singular Value Decomposition (TruncatedSVD) is a matrix factorization technique that factors a matrix similarly to Principal Components Analysis (PCA), excepting that the factorization for SVD is done on the data matrix, whereas for PCA, the factorization is done on the covariance matrix, to reduce the number of features.

C.4 Machine Learning Algorithms

Naïve Bayes (NB) classifier uses the Bayes Theorem to compute the probabilities of membership for each class, for each data instance. The highest probability is considered the most likely class.

K-Nearest Neighbors (KNN) algorithm is a non-parametric technique which uses a distance measure (normally Euclidean distance) to identify which are the k nearest neighbors of a document x among all the documents in the training set, and scores the category candidates based on the class of these neighbors.

Decision Tree (DT) is a classification method where the data is continuously split according to a certain parameter, in a process known as binary recursive partitioning. Data may be partitioned till the instances in the leaves are pure (instances at each leaf belongs to a single class) or earlier to prevent overfitting.

Support Vector Machine Classifier (SVC/SVM) is a method based on the idea of finding a hyperplane that best divides a dataset into two classes. It performs binary classification, so it is adapted for multi-class classification with the use of OvO (splits a multi-class classification into one binary classification problem per each pair of classes) or OvR (splits a multi-class classification into one binary classification problem per class) approaches.

Random Forests is an ensemble learning method that creates a multitude of decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.

Stochastic Gradient Descent (SGD) is a simple approach to fitting linear classifiers under convex loss functions such as (linear) Support Vector Machines. Strictly speaking, SGD is merely an optimization technique and does not correspond to a specific family of machine learning models. In the context of this work, it is equivalent to SVC.

XGBoost [Friedman, 2001] is an implementation of gradient boosted decision trees designed for speed and performance. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

C.5 Deep Learning Algorithms

MLP is a feed forward neural network that has no “state memory”.

CNN are feed forward neural networks that use a spatial-invariant convolutional kernel to efficiently learn local patterns, most commonly in images. They share weights across space and usually employ sequential convolutional layers.

RNN are networks that are cyclic in nature. They share weights across time, processing and efficiently representing patterns in sequential data. Therefore they have a “state memory”.

LSTM is a variant of RNN, that uses long short-term memory units as the recursive element.

Transformer is an architecture for transforming one sequence into another one with the help of two parts: encoder and decoder. The encoder takes an input sequence and maps it into a higher dimensional space (n-dimensional vector). That abstract vector is fed into the decoder which turns it into an output sequence.