CIA: Citizen Contact Center Agent Assistant

Luís Miguel Santos Neto
luis.m.s.neto@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2021

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.

1. Introduction

Since it was introduced to the world, email has become increasingly vital to our daily life. It has become a key tool for interpersonal communication and 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 [23, 2, 22, 13].

NLP is a field study of Artificial Intelligence (AI) that deals with the interaction between computers and humans using natural language. Most NLP techniques rely on ML to derive meaning from human languages, but 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 contact center of the Portuguese public services is a central channel of communication and interaction between the citizens or enterprises and the public services. About 1/5 of the communication throughput (an average of 500 daily incoming emails)[1] is based on emails and this amount is predicted only to increase. Despite being subject to a well-defined process of triage, prioritization, categorization and answer (described in fig. 1) the amount of emails that are received every day is too big, result in email overload and the process is not only repetitive, but contains repetitive tasks within its steps ('Read Emails').

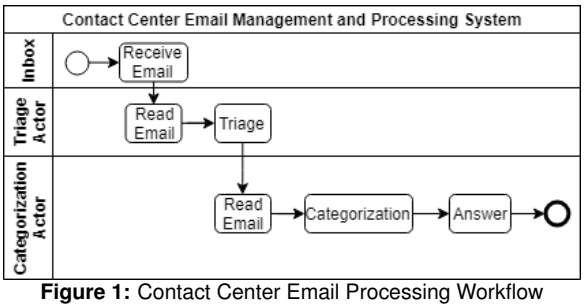


Figure 1: Contact Center Email Processing Workflow

In the 'Triage' step, the email is assigned to an agent (an agent that is expert in the subject of the email, the agent that dealt with the previous replies if it is not the first email, or the agent with fewer emails assigned). **The 'Categorization' step clas-**

[1] Data from the fourth quarter of 2019

sifies emails into predefined classes, organized in a hierarchy structure with the goal of assisting the 'Answer' step. In the 'Answer' step, a template email reply is selected from a set of reply template options, edited if needed, and sent. The set of templates is related to the category assigned: the further the specification of the category, the more specialized and enclosed the template set.

The goal of this work is to reduce human effort in the process represented in figure 1 by developing an automatic email classification system able to classify incoming emails, as represented in fig. 2.

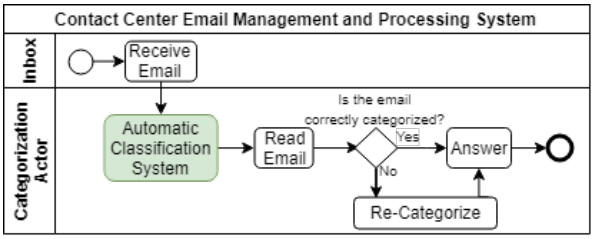


Figure 2: Desired Contact Center Email Processing Workflow

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.

Moreover, further contributions of this work may be summarized as follow:

- data quality: a study over the challenges of collecting raw data, forming a dataset, data cleaning and the impact of data noise in the automatic classification process performance;
- feature engineering: a report over the relation between feature extraction and selection techniques, classifiers and performance;
- a review of the application of several ML and DL strategies over a dataset with a particular irregular hierarchy of classes;

2. Background

The process of mapping a data instance with one or more classes from a predefined set of classes is called supervised learning. Classes may have no relationship (flat classes), have a ranking/ordering relationship (ordinal classes) or even be organized in a hierarchical structure/relationship (**hierarchical classes**). The class relationship/structure has a high influence on the approaches that may be employed during the classification process [17]. Furthermore, not only the structure of the classes affect the classification process, but also class issues such as **class imbalance**, **class overlap** and **label noise** have a huge impact.

Figure 3 summarizes the relationship between these issues, which are described in the following sections.

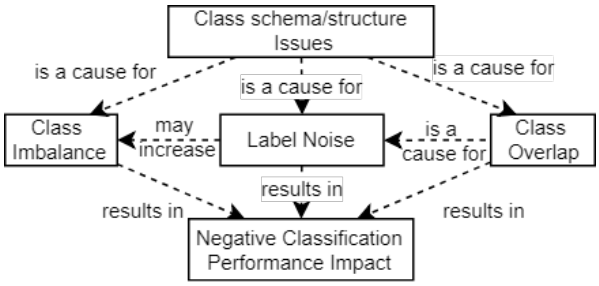


Figure 3: Class Issues Relations

2.1. Hierarchical Class Structure

Typically in large-scale data environments, there are huge amounts of data that 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. Asymmetry: if $m \prec j$, then $j \not\prec m$ for every $m, j \notin C$
2. Anti-reflexivity: $m \not\prec m$ 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 the 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.

Text classification problems may be categorized accordingly to fig. 4.

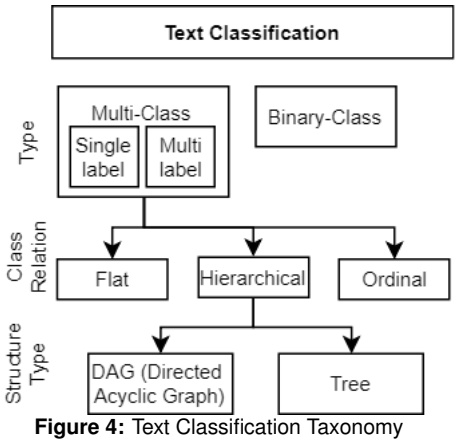


Figure 4: Text Classification Taxonomy

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

- A flat classification approach disregards class 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 the local approach builds a classifier per level, per node or per parent-node and the global approach learns a global single model for all classes.

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 these 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 [7, 1, 5] These techniques may be organized in a four type taxonomy:

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.

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.

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.

Ensemble-based approaches combine any of the previous strategies, namely external, with an ensemble-based algorithm.

2.3. Label Noise

Label Noise has been identified as one of the most problematic types of noise in the process of text classification [15, 6] and 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 relationship between the classes and the features [6].

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 performances. Consequently, the complexity of inferred models, as well as the number of required training instances, may increase [6]. In some domains, systematic label noise may distort observed results in a degree that may cause or intensify class imbalance problems.

2.4. Class overlap

In multi-class problems, especially 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 [11, 26]. This problem is called class overlap and is mainly caused by two factors: label noise (as mentioned in the previous section (2.3) and class structure/schema issues [28]).

Solutions to class overlap problems are usually based on the core idea of discarding, merging or separating the data instances located in the overlap region [26].

2.5. Category 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” [20]. A hierarchical class structure normally represents 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 [28].

“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” [27].

Hierarchies need to evolve, but they evolve at a much slower pace than data, and during this process, two major problems may arise[27, 28]:

Structure irrelevance: a hierarchy may characterize 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 be-

come less discriminative for the current data distribution;

Semantics irrelevance: semantics change in time and a category name that used to represent a collection of data may become less adequate, with the data it represents becoming more semantically related to 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 [27]. 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 works on this subject, a variety of approaches have been suggested: redesigning a class hierarchy based on concept analysis [24], changing relationships between categories [21, 27, 28] and editing the classes (remove/add classes) [28].

3. Related Work

Most research [13] is focused on email binary classification problems (Spam filtering and Phishing filtering), leaving only some scarce email multi-class classification articles of relevance. 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.

3.1. 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" [4]. 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 with "comparable performance".

Neural Networks (NN) were applied for the first time (to the authors best knowledge) at email multi-class classification in LINGER [3], "an NN-based system for automatic e-mail classification" that used an MLP (Multi-Layer Perceptron) trained with back-propagation, to classify five personal email datasets. The results showed that bigger proportions of emails to classes resulted in better performance and that for a single dataset, the choice of feature extraction and feature selection methods could cause an impact of 40.63% in performance. The feature extraction methods used were TF-IDF (Term Frequency Inverse Document Frequency), TF (Term Frequency) and TP (Term Presence), and the feature selection methods were IG (Information Gain) and Variance. Using the publicly available dataset of emails, Enron [8], Xia et al. [25] performed a classification experience (using ML methods) where SFS (Semantic Feature Selection), a method based on TruncatedSVD for reducing the dimensionality of large datasets and extracting the dominant features of the data, not only was able to drastically reduce the number of features but also improve accuracy and efficiency.

3.2. Email-similar Classification Systems

Contrarily to these studies, an automatic text classification system developed for IT incident tickets classification [18], studied the application of several techniques at the preprocessing phase, namely tokenization, stemming, elimination of stop-words, NER (Named-Entity Recognition), and TF-IDF based document representation, using SVM and KNN classifiers, concluded that the choice of document representation method, tokenizer, stemming or NER had little to no influence over the accuracy score. This experience classified incident tickets (which have a structure similar to emails), using ML methods, over a hierarchy of classes, with two levels (10 classes in the first level and 94 in the second level). As expected, accuracy was much lower when performing classification on the second level.

Similar to a contact center, the IT support help desk handles emails/messages regarding a set of subjects in a repetitive pattern. To automate this process, a classification system to categorize requests (emails) directed to the help desk was developed where emails are categorized with one category from each of the three levels of categories, containing 84, 8 and 77 unique categories [16].

The classifier model was built using 260.000 emails extracted from history. Each email was preprocessed, using OCR to extract text contents from the attachments, Microsoft LUIS to extract the intent, keywords extraction, 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 (Chi^2) scoring.

The classification process involved Random For-

est, XGBoost (ensemble learning), LSTM, Bi-LSTM, BERT and a hierarchical model built by some combinations of feature extraction - feature selection - classifier. BERT was excluded due to its high computational cost. To perform hierarchical classification, classes were divided into two groups: high accuracy classes and low accuracy classes, where low accuracy classes are represented as a single class in the former.

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 rules classification applied on 'body+title+OCR' with custom data engineered features with 76,5% F1 score was chosen due to being much less computationally expensive.

3.3. Hierarchical Text Classification Systems

In more broad text classification studies, some studies suggest the use of hierarchical deep learning (HDL) approaches. Most notably, HDLTex [9] was presented to perform local (top-down) hierarchical classification. The HDLTex architecture was built on top of three deep learning architectures (namely Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)) where for each level of the hierarchy, 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 an n-level hierarchy, HDLTex model is then composed of n DL sub-models. The best HDLTex model is chosen empirically, meaning that from an n-level hierarchy, 3^n HDLTex models are built in order to choose the best one. The models use different feature extraction approaches.

In contrast, the unified global hierarchical deep learning (Global HDL) classifier [19] 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 an 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 across different levels of classification. At each level there is a two layered MLP, which predicts the category at that level.

Both approaches for hierarchical TC were tested on the benchmark dataset *WOS-46985*, a document collection of 46985 documents labelled under a hierarchy of two levels. 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 [9] as were also FastText, Bi-LSTM with max/mean pooling and the Structured Self-attentive classifier [19]. HDLText shows superior performance at each level individually (90,45% and 84,66% vs 89,32% and 82,42%), but overall it is surpassed by the Global HDL model (76,58% vs 77,46%).

4. Data Context

4.1. Domain Understanding

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 to 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 to the email.

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.

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 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.

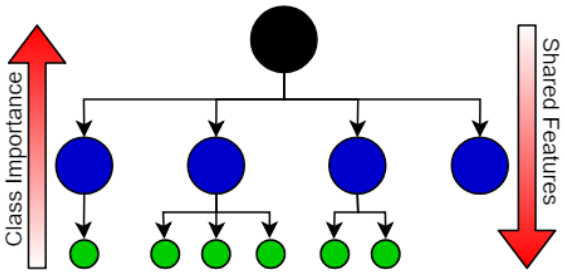


Figure 5: Hierarchy level importance/distinguishing power

Furthermore, data in these emails is of sensitive nature and had to be anonymized before it was delivered.

4.2. Data Exploration

All data collected and delivered for this experience was in raw format, organized in JSON files, each one representing a chain of emails exchanged between contact center agents and citizens, called "ticket" in the context of the contact center.

The process of interpreting and transforming the raw data in the JSON into a CSV dataset was hindered by two aspects:

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 analyze the data and required a manual and careful analysis of the data to distinguish the meaningful data from the useless data.

Anonymization: The anonymization strategies that were applied were mainly based on suppression of sensitive data which rendered data less readable and understandable.

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 to remove noise, standardize values and increase the percentage of useful, meaningful data. The dataset obtained at the end of this phase looses the suffix '*_raw*' and 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, there were three data acquisition 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 to achieve better results. Table 4.2 summarizes some basic insights of each dataset.

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

(NC = Noise Cleaning; DN = Data Normalization)

Table 1: Datasets Info

The anonymization process was softened at each iteration. At iteration 1 it removed all punctuation, replaced all named names with a placeholder and removed all numbers. At iteration 2 it stopped removing punctuation and at iteration 3 it started to replace all numbers with a placeholder instead of removing them.

Not only the anonymization process but also the data cleaning process evolved from iteration to iteration: at iteration 1 there was a manual process of noise identification and cleaning, at iteration 2 the manual process was replaced by an automatic process of noise identification and cleaning (performed before anonymization) and at iteration 3 a data normalization process was added to improve the meaning of relevant data. The result of these actions is reflected in table 4.2 and in figures 6 and 7.

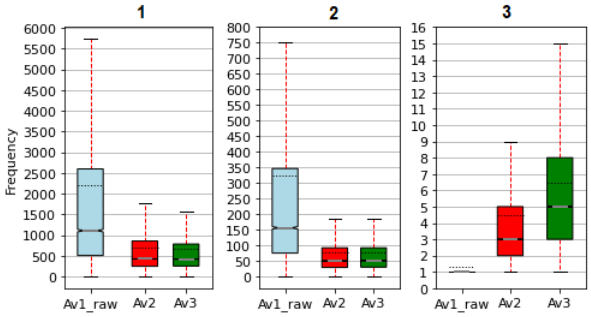


Figure 6: Evolution of the Average Frequency of Characters (1), Words (2) and Sentences (3) per Email by Dataset

The data noise identified 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 procedure.

The anonymization process (systematic noise) The anonymization strategies applied (which were decided by the contact center) were mainly based on suppression of sensitive data and impacted both noise and meaningful data, making it harder to distinguish between data and noise (even for a human).

Human writing (natural noise) Each chain of emails ('Ticket') is written by a different citizen, 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 young, and will show many misspellings, wrong words, ill-formed sentences, synonyms, different sentence structures for the same idea, etc.

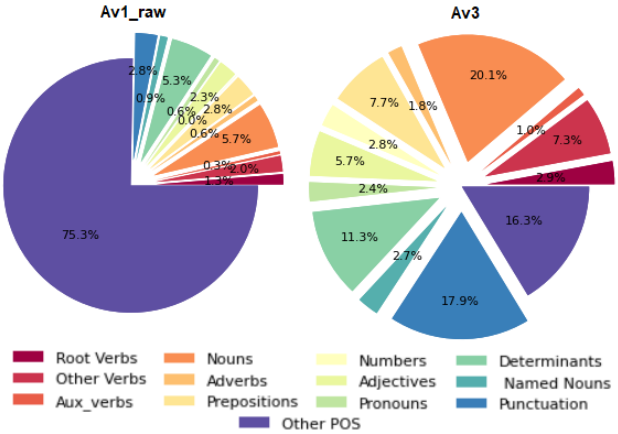


Figure 7: Evolution of Average Identified POS Tags per Email from the First to the Last Used Datasets

At iteration 3, strategies to clean/normalize data were usually centered around the application of regular expressions that parse all data and when they match a given pattern a transformation is applied. Regular expressions are simple to write, however, a

lot of attention has to be paid to their results because there is a large amount of data and the patterns that may be found within it are not entirely known. To ensure a successful application of regular expressions, their results were analyzed carefully almost as a data exploration task. This situation resulted in the discovery of new normalization opportunities. Figure 7 shows the evolution of the meaning extracted from each email, from the first dataset to the last.

4.2.1 Data Description

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

1. **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".
2. **Text** is a string representing the email body content. It is unstructured data and the main source of features.
3. **Subject** is a string representing the email subject.
4. **Ticket** is a string identifying a chain of emails. For a given ticket value the value of the subject is constant
5. **Reply Nr** is an integer representing the position of the email in the ordered chain of emails.

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. Figure 8 is an excerpt from the class structure. 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-node classes, making it very hard to discriminate between data instances assigned with child or parent classes when performing classification.

At the last iteration of development, dataset Av3 class structure is organized as follows:

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.

For business reasons or due to manual classification issues, not all the data extracted was classified

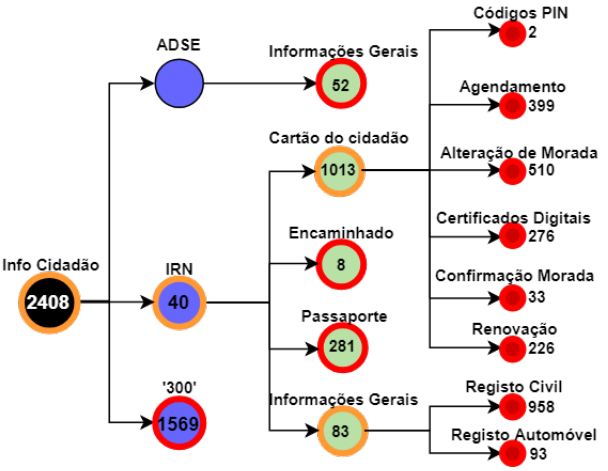


Figure 8: Excerpt of the class taxonomy including the frequency of files at each class (red = leaf node classes; orange = inter-level classes with emails assigned)

to the leaf node class. This issue and the solutions applied are explained in section 5.1.

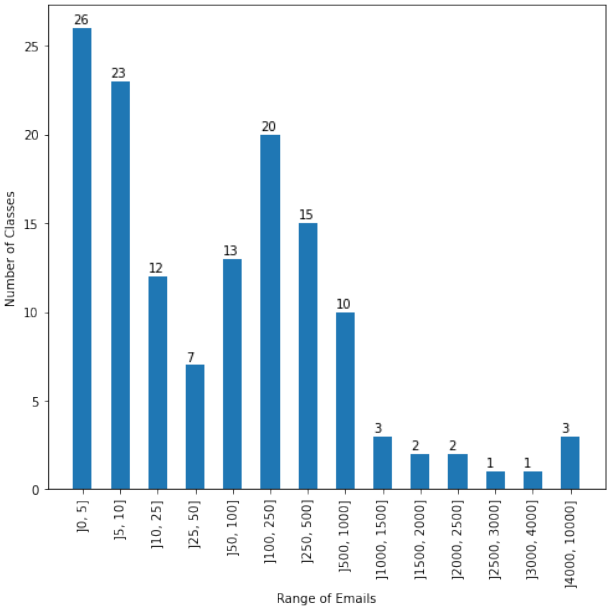


Figure 9: Histogram of Emails per Class (the value of the bar is the number of classes with a frequency of emails contained in the interval at the base)

The target classes for the classification process should only be the leaf node classes, which means that in a flat classification perspective we should have 120 classes.

The class taxonomy (fig. 8) reveals some semantic and structural problems and the class email distribution (fig. 9) portrays a serious case of class imbalance (with 26 classes with only 1 to 5 emails which represents a huge fraction of the classes that exist and that presents a big challenge to classification). 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 others 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 2.5) a child class inherits all the characteristics of its parents.

5. Development

5.1. Class Structure Inspection and Re-structuring

Our dataset, as may be observed in section 4 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 $49564/138 \approx 359$ 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 imbalanced 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 class instances for each of those classes. Furthermore, oversampling carries the risk of over-generalization 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, we propose and developed methods that take advantage of the hierarchical nature of classes to counter class imbalance and other issues described earlier in sections 2.2, 2.4, 2.3 and 2.5. The most notable methods developed are described as follow :

1. **Hierarchical Cut (hc$L$)**: Reduce all classes to level L of the hierarchy. All classes that are at a higher level than L are aggregated at L level.
2. **Balanced Hierarchical Cut (bhc$L$_Mmax):** 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.
3. **No Middle (inter-level) Class instances (nmc):** 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.
4. **No Only Childs (noc):** 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.
5. **Aggregate sibling classes in parent category (aggLx):** 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.
6. **Merge classes (mi):** 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.
7. **Remove classes with less than *n* instances o 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 with one another. For example, if we would apply the strategies of Balanced Hierarchical Cut at level 3 with maximum of 500 instances (bhc3_M500), followed by No Middle Classes (nmc) and Remove classes with less than *100* instances (m100) to Av3, we would produce an offspring dataset called Av3_bhc3_M500_nmc_m100.

5.2. Text Classification

The Portuguese language is highly inflectional, 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 [14] and RSLP [12] and the lemmatization was done using *spaCy*'s lemmatizer (Portuguese large model "pt_core_news_lg"[2])

During the feature extraction and feature selection steps, the features extracted from the text were done using most commonly TF-IDF, but also TF and TP (Term Presence). Chi^2, MI (Mutual Information) and TruncatedSVD were the most used features selection methods. 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 creation of custom features is a common approach to create features with high information gain in order to improve the performance of ML algorithms. The most relevant custom features that were created and used were the 'previous email category' (class of the previous email in the chain of emails), 'reply nr' (the ordinal number of the email in the chain of emails), 'categories keywords' (key-

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

words that are part of the class names) and 'text statistics'.

When working with Deep Learning models, word embedding models were used, both pre-trained models and locally trained models. When training models locally, we experimented with dimension sizes of 100 and 300 and window sizes of 3 and 5.

During the modeling phase, an extensive list of classifiers was tested. All of them are listed in table 2.

Classifier	Hyper-parameters
Multinomial Naïve Bayes	(default)
Decision Tree Classifier	(default)
K-Nearest Neighbors (KNN)	k = 3 by default and all odd k values between 3 and 55 on some tests
Support Vector Classifier (SVC) - One vs Rest (OvR)	(default)
SVC - One vs One (OvO)	(default)
Stochastic Gradient Descent (SGD) - OvR	(default)
SGD - OvO	(default)
Random Forest Classifier	(default)
GradientBoosting	(default)
AdaBoost	(default)
XGBoost Classifier	Custom implementation using XGBoost booster and hyper-parameters optimized using Random Search and experimentally. Best model chosen using early stopping.
HDLTex MLP (Multi Layer Perceptron)	Best model chosen using early stopping
HDLTex CNN (Convolutional Neural Network)	5 levels of complexity, best model chosen using early stopping
HDLTex RNN (Recurrent Neural Network using GRU)	Best model chosen using early stopping
RMDL [10]	(default)
LSTM (Long Short Term Memory	Obtained Experimentally
BERT	Obtained Experimentally

Table 2: Classifiers Used

6. Evaluation

During evaluation each of the major datasets (Av1, Av2 and Av3) 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 these combinations produced a multitude of evaluation results in the order of the tens of thousands, from which (due to this document limitations) the next subsections present only a small excerpt that translate the most important insights.

6.1. Flat Classification

Flat classification ignores the class relationships and performs prediction considering all leaf nodes as independents classes. Figure 10 shows that the results increase from dataset to dataset, which means that increasing data quantity and data quality benefits performance.

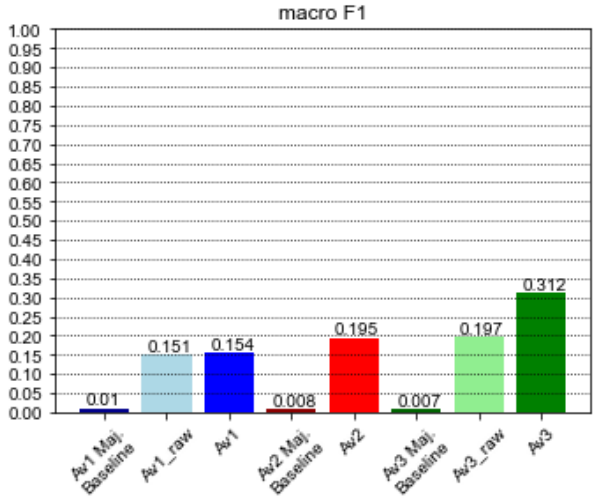


Figure 10: Evaluation results (macro-F1-measure) of datasets Av1 (blue), Av2 (red) and Av3 (green)

Figure 11 shows that class re-structure techniques described in section 5.1 improve macro-F1 performance mainly by reducing the number of classes with a very low amount of samples by removing them and by aggregating them and creating classes with more significance.

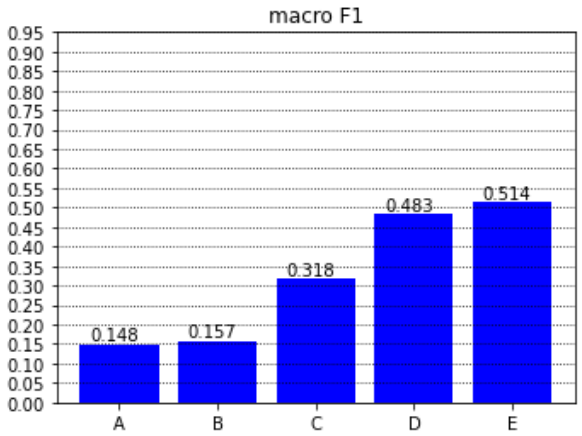
6.2. Local Classification Per Level Approach

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

We use the strategies of Hierarchical Cut and Balanced Hierarchical Cut, presented in section 5.1 to obtain class level-based datasets to train each classification model for each level. Figure 12 shows that just by aggregating balanced "cut" classes, using Balanced Hierarchical Cut increases macro-F1 measure, probably because the newly generated dataset classes have less class imbalance.

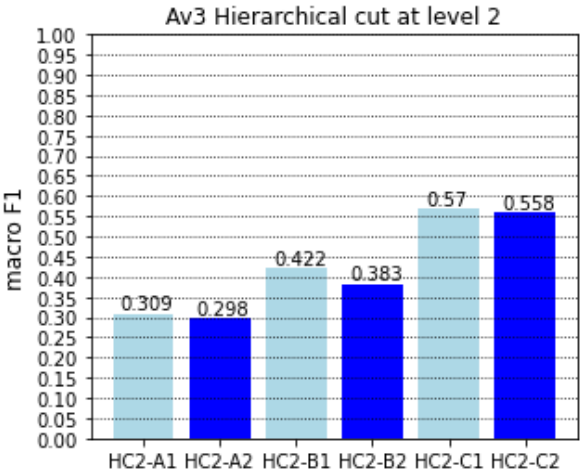
6.3. Local Classification Per Parent Node-based Approach

Classically, a local classification per parent node approach is a top-down approach where we train a model to classify each parent-node child-nodes. To



	Offspring datasets - Re-balance	Total Classes	Total Emails
A	Av1	105 (100%)	11894 (100%)
B	Av1_noc	98 (93,3%)	11894 (100%)
C	Av1_noc_ag100a_nmc	40 (38,1%)	9179 (77,2%)
D	Av1_noc_ag100a_nmc _mi100_m100	13 (12,4%)	8570 (72,1%)
E	Av1_noc_ag100a_nmc _mi100_m100_u100	13 (12,4%)	1690 (14,2%)

Figure 11: Evaluation results of AV1 offsprings



	Offspring datasets - Re-balance	Total Classes	Total Emails
HC2-A1	Av3_hc2	49564	25
HC2-A2			
HC2-B1	Av3_bhc2_M500	6259	24
HC2-B2			
HC2-C1	Av3_bhc2_M500	6073	15
HC2-C2	_mi100_m100		

Figure 12: Evaluation results (macro-F1-measure) of hierarchically cut Av3 at level 2

further explore the hierarchy relations and the semantic relations of classes, we developed artificial nodes/classes (represented by squares in fig. 13) to separate sibling classes to different classification models.

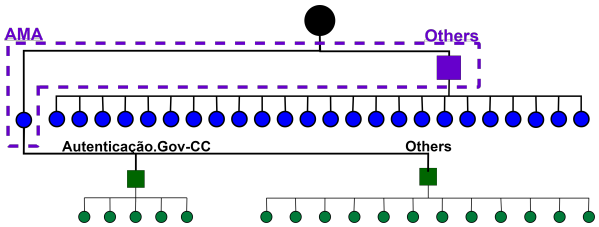


Figure 13: Class Structure Re-organization for Classification

Class	Nr. of Child Classes	Nr. of Emails
Info_Cidadão (root)	2	47156
AMA	2	30498
Others	23	16658
AMA/Autenticação.Gov-CC	5	25911
AMA/Others	12	4587

Table 3: Class Structure Re-organization for Classification - Class Info

These new structure with artificial nodes (displayed in fig.13 and described in table 3) was defined taking in consideration the semantic relations of classes, class-email frequency distribution and the structure of the hierarchy. The structure was found empirically, with the support of clustering techniques and semantic analysis and built with the help of the re-structuring methods presented in section 5.1.

Model	Classifier	L. ma-F1	L. Acc	O. Acc
AMA vs Others	XGBoost	0.725	0.761	0.761
AMA/Autenticação .Gov-CC vs AMA/Others	SGD (OvO - OvR) / XG-Boost (tied)	0.912	0.913	0.695
AMA/Autenticação .Gov-CC	OvR - SGD	0.81	0.872	0.606
AMA/Others	XGBoost	0.788	0.866	0.602
Others	XGBoost	0.579	0.631	0.458

L. Acc = Local/Level Accuracy; O. Acc = Overall Accuracy

L. ma-F1 = Local/Level macro-F1

Table 4: Classification Performance for Parent Node-based Approach

Local Accuracy represents the accuracy of the local model, while overall accuracy represents the accuracy of the composed model, to classify an email from the root to a given level. Local performance is much improved, compared to the hierarchical models presented in section 6.2, and while overall accuracy is reduced, it is superior to the accuracy obtained with other models.

7. Discussion

Flat classification does not seem the most adequate for this kind of problem. It puts aside the hierar-

chy relations which are important to structure data and even when applied with class re-structure techniques, that sacrifice lots of data and classes to obtain better results, performance is still limited.

Local Classification approaches are more promising. With per level approaches, classifying data only to hierarchically cut classes at level 2 presents the best obtainable results and assigns the most important classes to data. Further classifying data to level 3 or 4 results in 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.

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 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 12), using TF-IDF features. XGBoost achieved the best performance with almost all the hierarchical approaches and most offspring datasets.

8. Conclusions

The main objective, declared in section 1, 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.

Our contributions revolve around reporting the use of strategies regarding data cleaning, data normalization, feature extraction, feature selection, dimensionality reduction, feature engineering, text classification using ML, text classification using DL state of the art approaches (BERT), 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. But the main contribution of this work is the proposal of new class re-balance strategies, that instead of undersampling or oversampling data, explore the structure and semantics of a hierarchy of classes and reorganize it in order to optimize classification performance and keep hierarchy semantic meaning.

To the best of our knowledge, there are no works in literature about hierarchical classification of real-world data with systematic and natural noise on irregular and problematic hierarchies such as the one that grounds this work.

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.

References

[1] H. Ali, M. Salleh, R. Saedudin, K. Hussain, and M. Mushtaq. Imbalance class problems in data mining: A review. *Indonesian Journal of Electri-*

cal Engineering and Computer Science, 14, 03 2019.

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

[3] J. Clark, I. Koprinska, and J. Poon. A neural network based approach to automated e-mail classification. pages 702 – 705, 11 2003.

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

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

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

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

[8] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. volume 3201, pages 217–226, 09 2004.

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

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

[11] S. Lavanya, D. S. Palaniswami, and S. Sudha. Efficient methods to solve class imbalance and class overlap. 2014.

[12] V. Moreira and C. Huyck. A stemming algorithmm for the portuguese language. pages 186– 193, 12 2001.

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

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

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

[16] K. Shanmugalingam, N. Chandrasekara, C. Hindle, G. Fernando, and C. Gunawardhana. 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, 2019.

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

[18] S. Silva, R. Ribeiro, and R. Pereira. Less is more in incident categorization. 07 2018.

[19] K. Sinha, Y. Dong, J. C. K. Cheung, and D. Ruths. 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, Oct.-Nov. 2018. Association for Computational Linguistics.

[20] G. Snelting and F. Tip. 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, 1998. Association for Computing Machinery.

[21] L. Tang, J. Zhang, and H. Liu. Acclimatizing taxonomic semantics for hierarchical content classification. volume 2006, pages 384–393, 01 2006.

[22] I. The Radicati Group. Email statistics report, 2015-2019. Technical report, Mar. 2015.

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

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

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

[26] H. Xiong, J. Wu, and L. Liu. Classification with classoverlapping: A systematic study. pages 303–309. Atlantis Press, 2010/12.

[27] Q. Yuan, G. Cong, A. Sun, C.-Y. Lin, and N. Thalmann. Category hierarchy maintenance: A data-driven approach. 08 2012.

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