Improving telecommunication customer service through text analysis and categorization using semi-supervised learning

Maria Eusébio
mariasdeusebio@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

September 2021

Abstract

With the increase of customers expectation, specially in telecommunication services, and being customer retention less expensive than customer acquisition, companies are investing more time and money on improving customer service, by automating internal processes and thus having a faster and more efficient troubleshooting process.

This work explores some methodologies that can be applied in the context of telecommunications and call centers, namely in the areas of text mining, Natural Language Processing (NLP), machine learning and Semi-Supervised Learning (SSL) algorithms, such as Label Propagation and Label Spreading, since the data provided is mainly unlabeled.

The primary objective was to classify each troubleshoot guide text according to each variable involved in the diagnosis of the recommendation system, as separate text categorization sub-problems, and from that build a new dataset with all the possible sequence of events and the correspondent troubleshoot guide. However, since the text categorization results obtained with the SSL algorithms were not as good as expected for some of the sub-problems, an alternative solution was found, consisting in, first, cluster the original labels into smaller sets and then classifying the texts into each of them. The solution obtained could be used to narrow down the huge amount of possibilities, helping the technical specialist in the task of associating the texts to the correct label in each sub-problem and, in that way, improving the existing recommendation system.

Keywords: Natural Language Processing, Text Mining, Text Categorization, Semi Supervised Learning, Telecommunication Services

1. Introduction

Troubleshooting telecommunications problems involves a quite complex process from the received call until the decision on the best solution since, as stated above, there are many possible fields and each has its own specificities that, depending on the circumstances, may also vary. Besides the data obtained from the costumer's complaint, used to perform a diagnosis of the problem, troubleshoot guides are also used to know which process to follow next.

NLP and Machine Learning (ML) techniques will be applied to the available text data, more specifically, text classification, where one or more classes from a given set of possible classes is assign to a text document, in order to automate some of the technical assistant tasks. Since obtaining a considerable large labeled dataset, necessary for the text classification task, is typically very expensive and requires a substantial human effort, the SSL, introduced in the 1960s, algorithms are explored in this thesis.

This research aims to improve an already implemented recommendation system used in the technical support service of a telecommunication company, namely through developing automated solutions capable of increasing the effectiveness in the process of diagnosing the telecommunication issues that are reported to the company's technical assistants by its customers.

1.1. Problem Setting

The process starts with some kind of failure in a device or service, that is then described by the costumer when he calls to the technical assistant. Meanwhile, the recommendation system collects information about the costumer's situation (a combination of components), the Scenario, that will be used by the technical assistant during the call, to narrow down the existing possibilities. The technical assistant has also to select other components in the recommendation system, based on the costumer's complaints, namely the Service (Internet, Televisão or Voz) in question and the Symptom related to the complaint. In each of them there are more specific options that should also be selected, Sub-service and Sub-symptom, respectively. With this information the recommendation system leads to the 3 most probable Causes, from an existing list of possible causes, for the costumer's complaint.

After deciding which is the actual cause of the reported issue, the technical assistant follows some procedure according to the detected cause, in order to solve that specific problem. The list of all possible procedures (troubleshoot guides) is stored in a database out of the recommendation system, that has explanatory texts with the steps to follow after finishing the diagnosis in the recommendation system. The aim was then to bring together the information in the recommendation system and in this troubleshoot guides, so that the process could be even faster and more efficient.

1.2. Literature Review

This research is centered in text analysis and thus it was necessary to first explore the many specificities of this unstructured data, starting with the main notions of text mining and NLP ([9]). Then all the steps involved in the process of text classification were addressed ([10]).

The process starts with the text preprocessing, which includes a preliminary text cleansing, tokenization, Part-of-Speech (POS) tagging and word normalization (a comparison between stemming and lemmatization approaches, as well as their limitations, is held in [18]). It follows the text transformation, a crucial step since text is unstructured data, that consists of transforming text into numerical data so that it can be used in the classification algorithms. There are essentially two kinds of approaches to perform this transformation, by producing one number per each word (wordcount or bag-of-words, [15, 16]) or by producing one vector per each word (word embedding, [1]).

One of the most problematic issues of text categorization is the high dimensionality of the feature space, since it consists of the unique terms (words or phrases) that occur in documents and that may correspond to tens or hundreds of thousands of terms, even for a moderate-sized documents collection. A simple approach, called feature selection, consists of removing the non-informative terms according to documents collection statistics, e.g. Chi-Square (Chi2) e and Mutual Information (MI) ([21, 20]). Another possible approach, feature extraction, constructs new features, combining lower level features (words) into higher level orthogonal dimensions. The most common technique is Principal Components Analysis (PCA) ([8]), but other methods have been studied in the text categorization problems, such as Latent Semantic Analysis (LSA) ([17]) or Latent Dirichlet Allocation (LDA) ([4, 14]).

With the data prepared, there can be then applied the text categorization techniques, where the goal is to classify the documents into a set of categories. Ideally, this would be performed by supervised learning techniques, such as the Naive Bayes (NB) classifier ([3]), however, this requires an extra human effort in order to pre-define the categories and assign them to the documents of the training set. The unsupervised leaning techniques try to overcome this issue using clustering algorithms, such as K-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) ([13, 7]).

The idea of combining both supervised and unsupervised learning, aiming to make use of the unsupervised approaches as a first hint to enhance the supervised learning tasks, has then appeared ([11]), resulting in the SSL approach. These techniques can be categorized into self-labeled algorithms, generative models, low-density separation algorithms, graph-based methods ([5]), but this work will mainly focus on the latter, namely the Label Propagation (LP) and Label Spreading (LS) algorithms. The simplest SSL technique, self-learning ([19]), as well as a generative model, Expectation Maximization (EM) algorithm ([6]) are also implemented for comparison.

Since the provided dataset presents a clear class imbalance, a common issue in multi-class problems, the performance of the text categorization algorithms is evaluated with accuracy and weighted F_1-score ([12]). The performance of the clustering algorithms is evaluated with a metric defined according to the particularities of this problem and based on [2], where a few intuitive formal constraints on text clustering algorithms evaluation metrics are defined, leading to the most important aspects on the clustering quality.

2. Text Mining

2.1. Text Preprocessing

When working with text data, the observations will be a set of documents (document collection or corpus) and its features are usually the words deriving from each document, hence, the quality of the text mining operations depends on the data preprocessing methodologies applied.

Typically, the first step in this stage is to apply tokenization, obtaining a set of terms (tokens) for each document, that can correspond to a single word or a set of n consecutive words ($n-$gram). Then, it is removed the set of most common words (stopwords, e.g. "e", "que", "um", "uma", etc. in

Portuguese) since they add no relevant information and, moreover, punctuation and numbers are also removed. Additionally, spelling correction is almost always applied in order to identify and rectify some possible handwriting errors, typos or even abbreviations, using a certain dictionary.

The POS tagging is used to associate each word with a suitable category (semantic content) based on their function (and position) in the sentence, e.g. Article, Noun, Verb, Adjective, Preposition, Number and Proper Noun. Furthermore, a morphological analysis can also be done, through word normalization. The two main approaches are stemming, that obtains the radical or stem (approximation of the base form) of each word by removing suffixes and/or prefixes that do not correspond to its base form, and lemmatization, that obtains the canonical/base form (lemma) of a word by completely removing the suffix/prefix of a word or replacing it with a different one. This will substantially reduce the number of features, with the idea that words with similar base forms have similar meaning and thus can be processed as the same word.

2.2. Text Representation

Since the known learning algorithms are used to process numeric kind of data and not words, it is necessary to transform the texts into some machine readable data and so the texts are usually represented by feature vectors, being each text a sequence of features and their weights. The simplest and most common way of representing these weights is through a Bag-of-Words (BoW), that is, a document term matrix, where each row represents a document, each column represents a term and each matrix entry corresponds to the weight of the feature in that document. Typically word frequency (number of times each word occurs in the document) is used, but since higher weights are assigned to the most common (and usually not relevant) words, another technique is introduced in order to overcome this issue. The Term Frequency-Inverse Document Frequency (TF-IDF) model takes into account not only the word counts in the document but also the frequency of that word in the whole document collection, penalizing the most common words. The weight of word w in document d is then given by $W_{TF-IDF}(w,d) = TF(w,d) \times \log\left(\frac{m}{df(w)}\right)$, where $TF(w,d)$ is the frequency of word w in the document d, $df(w)$ is the number of documents that contain word w and m is the total number of documents.

In order to also include the meaning of words and the similarity between them, word embedding techniques have been implemented as well, where words are represented by vectors of real numbers in a high dimensional space, based on their context (neighboring words). Word2Vec (W2V) is one of the most common algorithms and consist in training a neural network, with only an input layer, a projection layer and an output layer, in order to learn and predict the vector representation of each word. The architecture is based on either a Continuous Bag-of-Words (CBOW), that predicts a target word from one or more context words, or a continuous Skip-Gram (SG), predicting one or more context words from a target word. Since this algorithm does not take into account the actual context of the word, as each word has only one vector representation, there was the need to introduce the contextual embedding methods, that consider the whole sentence as the input, learning more than one vector representation for each word. Bidirectional Encoder Representations from Transformers (BERT) is one example of these complex language representation models, where both the left and right (bidirectional) context of each word is taken into consideration, making it possible to pretrain a deep bidirectional Transformer, thanks to a Masked Language Model (MLM).

2.3. Dimensionality Reduction

In order to handle the inherent high dimensions of the text data, there are some dimensionality reduction techniques that may be applied, reducing the n dimensions of the feature space to k, with $k \ll n$. One way of doing it, called feature extraction, is to build a new set of features, based on combinations of the original feature set, through algorithms such as PCA, LSA or LDA. PCA and LSA are both based in the Singular Value Decomposition (SVD) (in PCA some normalizations are made before), however, LDA is a bit more complex, as it builds a generative probabilistic model of the corpus with the help of Gibbs Sampling, where a set of latent topics will work as "bridges" between the documents and the words.

Another possible approach is to simply remove irrelevant words, the feature selection.

The Chi2 is one of the most popular feature selection methods used in text classification and it tests the independence of two events, occurrence of the word and occurrence of the class, measuring the lack of independence between them, based on the χ^2 distribution. The k words with the highest total Chi-square values are selected and the remaining words are discarded. On the other hand, the MI scores are based on the entropy, measuring how much information the presence or absence of a specific word contributes to making the correct classification decision on a certain class.

3. Text Categorization

Although the task of classifying documents into a defined set of categories (classes) is usually done through supervised learning techniques, such as the well known NB algorithm, the unsupervised learning approach or even a combination of both (the SSL) can also be used to perform this task. Ideally, $X = (x_{ij}) \in \mathbb{R}^{m\times n}$ contains the m observations with the n associated features and, except for the unlabeled approach, there exists a vector $\mathbf{y} = (y_i)^T \in \mathbb{R}^m$, containing the labels of each observation, from a set of L possible classes.

Two possible clustering algorithms used in the unsupervised approach are the K-means and the DBSCAN. Whereas K-means is a distance-based partitioning algorithms, that starts by defining a set of central points around which the clusters are built, DBSCAN is a density based clustering algorithm, that identifies clusters based on the idea that within each cluster the density of points is significantly higher than outside of the cluster.

3.1. Semi Supervised Learning

The most suitable approach to use when the majority of observations from the dataset are unlabeled but there are a few observations with labels associated is the SSL. There are several possible forms of SSL but the one used in this research is the one that works like supervised learning, with additional information on the distribution of the observations (the unlabeled data), having the same goal of predicting a label for each observation (inductive learning). The dataset X is divided into $X_l := (\boldsymbol{x}_1, ..., \boldsymbol{x}_l)$, associated with labels $\mathbf{y}_l := (y_1, ..., y_l)$, and $X_u := (\boldsymbol{x}_{l+1}, ..., \boldsymbol{x}_{l+u})$, without labels associated, where l and u are the number of labeled and unlabeled observations, respectively, and obviously $l + u = m$.

In order to actually add relevant information through the unlabeled data, there are three assumptions that should be satisfied:

Smoothness assumption: If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2. This basically means that the label function is smoother in high-density regions than in low-density regions, that is, if two points are linked by a path of high density (e.g., if they belong to the same cluster), then their outputs are likely to be close, however, if they are separated by a low-density region, then their outputs are not necessarily close.

Cluster assumption: If two points are in the same cluster, they probably belong to the same class. This means that, in general, observations that belong to distinct classes do not appear in the same cluster, but it does not say that each class forms a single compact cluster.

Manifold assumption: The (high-dimensional) data lies (roughly) on a low-dimensional manifold. This is an important assumption in a sense that it prevents the so called curse of dimensionality, since in this case (if the data lies indeed on a low-dimensional manifold) the classification algorithm can function in a lower dimensional space (the manifold).

Regarding the types of algorithms used in SSL, the simplest one is Self-learning, that starts by training the a supervised learning model with the labeled data and, with that model, predicts the labels of some unlabeled observations using the current decision function. Then, the model is retrained with the labeled data and the new labeled observations and this process is repeated until all the unlabeled observations are associated with some label.

On the other hand, the EM algorithm is based on the NB classifier and it starts by defining it using the labeled data. Then, with this classifier, the unlabeled data is classified but instead of trying to find the most likely class it is calculated the probabilities associated with each class. These last estimated class probabilities are used as the true class labels for its data observations and a new NB model is build. Until the model converges to a stable classifier and all the data is labeled, the procedure of classifying the unlabeled data (**E** step) and rebuilding the classifier (**M** step) is iterated.

The graph-based methods, where the data is represented by a graph, in which the nodes are the data observations (both labeled and unlabeled) and the edges represent the pairwise distances or similarities between them, will be the focus here. The most popular techniques is the LP, where the idea is to propagate labels on the graph, starting with the labeled nodes, X_l, and their correspondent set of labels, Y_l, and propagating them to its neighbors through the graph edges (iterated this process until convergence).

The weight matrix is defined as the Gaussian kernel of width σ, that is, $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$. Moreover, the probabilistic transition matrix $T = D^{-1}W$, where T_{ji} represents the probability of going from node j to node i, with $T_{ji} = p(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{m} w_{kj}}$, and D is the diagonal matrix given by $D_{ii} = \sum_j W_{ij}$, is also defined. Then, the initial set of labels is defined, $\hat{Y}^{(0)}$, with the unknown labels set as -1, and a new vector of estimated labels is obtained by multiplying the transition matrix T with the previous vector, $\hat{Y}^{(t+1)} \longleftarrow T\hat{Y}^{(t)}$, repeating this process until convergence.

Another similar algorithm is the LS that, as the LP algorithm, starts by defining the affinity matrix W, with the slight difference that now each element of the diagonal is set to zero (to avoid self-reinforcement) and the matrix is normalized symmetrically for the convergence of the next iteration.

The next step is a loop where, at each iteration, each node collects information from its neighbors and, at the same time, preserves its initial information, $\hat{Y}^{(t+1)} \longleftarrow \alpha S\hat{Y}^{(t)} + (1-\alpha)\hat{Y}^{(0)}$. The parameter α corresponds to the relative amount of information that is obtained from the neighbors and from the initial information. Then, when labeling the unlabeled observations, the class of which an observation has received most information through the whole iteration process is the chosen label for it.

3.2. Evaluation Metrics

There are many possible ways of evaluating the generalization performance of a classifier, such as the F_β-score and the accuracy. For multi-class classification problems, some adjustments need to be performed to the original formulas, through averaging the evaluation metric results over all classes (macro average), M, or using the grouped results, that is, cumulative values (micro average), μ. Whilst macro-averaging treats all classes equally, micro-averaging favors bigger classes. The accuracy is given by the fraction of correct predictions over the total number of observations. Note that, thanks to the β value, that represents the ratio between the cost associated with predicting false negatives and false positives, F_β-score is most suitable for imbalanced datasets, since usually miss-classifying minority classes is more expensive than miss-classifying majority classes and so the F_β-score will be more (or less) affected when improving the recall rather than the precision for $\beta > 1$ (or $\beta < 1$).

On the other hand, evaluating the performance of a clustering algorithm is not so straightforward. The clustering evaluation metrics can be defined as intrinsic, that measure how close to each other the observations from the same cluster are and also how distant they are from other observations in the other clusters, or extrinsic, that compare the outputs of the clustering algorithm with some gold standard that is typically defined with human assistance (usually used in text clustering). Moreover, there exists a proposal of formal constraints for the latter:

Homogeneity: the clusters must be homogeneous, that is, each cluster should not mix observations from different categories.

Completeness: the observations from the same category should be kept together in the same cluster.

Rag bag: it is better to have a disordered cluster among the others, with all the observations that cause disorder, than having disorder spread by all the clusters. This cluster is the “rag bag”, sometimes called “miscellaneous” or “other”, and contains all the observations of diverse categories that could not be assembled with other observations.

Size vs quantity: it is best to have a bigger cluster with small errors than to overly divide the data into several small clusters with minimal associated errors, which will probably lead to overfitting.

4. Implementation

The computational tool used throughout this thesis was Python.

4.1. Dataset Description and Exploratory Analysis

The data provided for this problem was divided into 3 datasets, one with information on records from the existing recommendation system, other with some of the troubleshoot guides texts and the correspondent links (to where they can be found) and another one with a more specific description on one of the variables from the first dataset, that was only characterized by a code key, mapping it to each key (`CAUSE_KEY` and `CAUSE`). All these datasets were merged together and the columns considered irrelevant were removed, obtaining a final dataset with 85032 observations and 9 response variables, from which only 6 will be considered: `L0`, `L1`, `L2`, `L3`, `CAUSE` and `TEXT`.

Then, a first analysis on the response variables, that would be used in different sub-problems, was made in order to study how many non `NaN` values existed in each column (which was different, from 513 to 641) and from those values how many were unique values (i.e. number of labels), which was the most frequent value and how many times that value appeared. Whereas `L0` had 3 unique labels associated, `L1`, `L2`, `L3` and `CAUSE` had 14, 39, 36 and 92, respectively. Moreover, the frequency distribution of each class, for each of the 5 sub-problems was analyzed and it was observed that all of them had class imbalance. Also, in sub-problems `L2`, `L3` and `CAUSE` most of the classes had less than 10 observations (texts) associated to them, being 15, 13 and 33, respectively, unique observations (labels with only one linked text).

4.2. Text Preprocessing

First, the irrelevant information was removed from the texts, such as punctuation, URLs, e-mail addresses, numbers, unusual characters or symbols not included in the punctuation set and the resulting extra whitespace characters. Then, a dictionary was build, based on the Portuguese dictionary and including the acronyms, the English words and also telecommunications brands and problem specific words found in the texts and, with that, a spellchecker was defined in order to correct misspelled words in these texts and reduce classification errors.

After this cleansing, the Portuguese stopwords were removed. Noticing that the word “não”, that gives negative connotation to the its immediately following words, was also being removed, it had to be removed from the stopwords list and, moreover, the word “não” was joined with its immediately following word using an underscore, wherever it was found (to maintain negative connotation). Then, since there are still not many options for the NLP analysis for the Portuguese language, and the lemmatization generated quite bad results(e.g. mixing up nouns and adjectives), it was decided to only apply lemmatization to the text verbs.

4.3. Text Representation

In order to represent the texts by numerical values, the TF-IDF, term frequency (TF) and W2V techniques were applied. Since the embedding obtained with W2V is from only one word it was necessary to extend this idea to an entire document. For that purpose it was used an extension of W2V, Doc2Vec (D2V), that instead of computing embedding vectors for each word does that for each document. Another approach was to take the mean of each word vector representation obtained with W2V, use it as a weight for that word and then replacing the TF-IDF weights for the W2V ones in the correspondent words defined in the feature matrix built with the former method. Although BERT seemed to be the best and most complete option for text representation, it turn out to be even more computationally intensive than it was expected, and so it was not possible to run it for all the documents in the dataset (only 6% of the observations, besides being very time consuming - almost 1 hour with the simplest input parameters), so it was decided not to include this method in the experiments.

4.4. Dimensionality Reduction

The methods used for dimensionality reduction were PCA, LSA, LDA, Chi2 and MI. With PCA it was possible to choose the number of components to keep based on the amount of variance that needed to be explained, e.g. 95%, and not just by the number of components itself, being also a starting point for LSA and LDA. However, besides being sometimes necessary to first scale the data before applying PCA, the number of components in PCA cannot be higher than the number of samples, which generates different parameters depending on the amount of unlabeled data used.

Whereas PCA and LSA were both very fast and had similar results with the same parameters, LDA took a lot more time training its model and to obtain close results to the other 2 algorithms the number of components needed to be considerably reduced.

The Chi2 and MI algorithms had similar (almost always the same) results and were much faster to define then the feature extraction methods, with the particularity that only the MI can deal with negative values in features.

4.5. Model Fitting

The NB classifier was used as the baseline model for this problem (although only applied to the labeled data). Then, trying to include all the provided data (including the unlabeled part), the SSL algorithms were applied, however, the LP and LS required a high computational capacity and thus it was only possible to run these models using at most 30% of the unlabeled data. Hence, the Self-Learning, the EM algorithm, and the LP and LS algorithms were applied using the whole labeled subset and 0%, 10% or 30% of the unlabeled data.

Moreover, probably related with the considerable imbalance between the classes in the labeled set and also the quite large amount of different possible labels, the first results were not as promising as expected. In fact the unlabeled data seemed to harm the results or not having influence at all. Since it was not possible to obtain sufficiently good accuracy results when trying to classify the texts into a specific label, the initially stated goal of this work had to change and the aim was then to classify the texts into a small set of labels. For that purpose, the experiences are divided into 3 different scenarios. The scenario 1 was already explained (uses only the SSL algorithms), the scenario 2 mixes both unsupervised (K-means and DBSCAN clustering algorithms) and SSL methods and the scenario 3 uses only the unsupervised learning techniques (to understand if the SSL stage used in scenario 2 was not a redundant step).

In scenario 2, the labeled data is first clustered, obtaining new labels (the clusters) and then the SSL algorithms are applied as in the scenario 1. For the scenario 3 only the labeled data and the K-means (the only one with the predict functionality) algorithm are used, so that the clustering can be evaluated as if it was a classification algorithm.

Since the 5 sub-problems were dependent on each other and should be linked in a specific order, in order to include the information of the precedent sub-problems it was decided to include the name of the labels provided by each of those sub-problems as words (features) in the texts of the current sub-problem. For instance, when classifying the texts into the sub-problem `L1`, the known `L0` labels associated with those observations should be appended, as a new word, to the texts used as input data set and this should be repeated for each data set observation (text). For the classification regarding the `L2` sub-problem, both `L0` and `L1` should be added to the input texts, and so on, until the last sub-problem, related with the `Cause` labels, is reached.

Hence, the workflow was defined by a set of consecutive steps. First, for each of the 5 sub-problems, the vector of target values, y, had to be obtained, encoding the text labels into numerical ones and setting the unlabeled observations to the label '-1' (the identifier used for the unlabeled observations in the SSL algorithms). The feature matrix X was always the same for all sub-problems, obtained from the column `TEXT` of the main dataset. Then, the feature matrix and the labels vector were separated into labeled and unlabeled and a small percentage of the unlabeled data was set to be used further on. The labeled data was split into train and test sets (70% and 30% of the labeled dataset, respectively) and then the subset of unlabeled data obtained before was introduced in the train set. Finally, the SSL model was defined and fitted to the train set. With the test set, the model was tested and evaluated through a classification report, the accuracy and the weighted F_1-score.

Note that, for the scenarios 1 and 3, trying to solve the class imbalance and its impact on the results, the unique observations were only added to the train set after the train/test split. The models were first tested without this mechanism but, as the results were greatly worse, it was decided to use this henceforth.

In order to capture the particularities of the given problem, a specific performance metric was defined to evaluate the quality of the clustering, considering essentially 4 (or 5, for the DBSCAN) parameters:

Number of clusters: A smaller number of clusters is associated with a simpler model, yet this number being higher to the original number of classes is not an absurd idea and this will be observed in the results.

Number of repeated clusters: If there are too many repeated clusters, that is, clusters associated with exactly the same labels, it indicates that the data set is being excessively partitioned and this could lead to overfitting.

Number of clusters with more than 4 labels: The number 4 was the initial choice, but then it was determined that this number could go up to 10. Here the idea was to narrow the hypothesis that would be later analyzed by the specialized technician, that was the main goal of these scenarios.

Maximum cluster size: This parameter is used in order to balance the previous one. If there are a lot of clusters with more than 4 associated labels but this number does not overcome a cap of 10 or even 20 then this is not problematic. On the other hand, if there are clusters with a considerable high number (e.g. more than 20) of associated labels it probably means that this is not the best cluster distribution. *Noisy observations (cluster '-1'):* Only relevant for the DBSCAN algorithm. It was first used to check if it existed the cluster '-1', which was initially considered as a drawback, since it meant that those observations and their associated classes were not being associated with any cluster. However, it was observed that the labels that were being attributed to this special cluster were exactly the ones with only 1 or 2 observations and so it was expected that they could not be correctly categorized in any of the other clusters.

5. Results & Discussion

5.1. Results

Each of the 5 sub-problems was analyzed separately, starting with the TF-IDF representation for the matrix X and testing all the possible models with the different amounts of unlabeled data. Then, if the results need to be improved, the dimensionality reduction techniques are applied and, afterwards, other text representations are also tested and the results compared.

In the sub-problem `L0` the results with the TF-IDF representation were quite good, above 0.9387 for NB, LP and LS, having Self-learning and EM worse results. The LP is the only SSL algorithm that overcomes the baseline model (with a 0.9793 score), but the difference is less than 1% and it was only able to correctly classify the majority class. Also, the unlabeled data used as no effect in both LP and Self-learning results.

The results with other text representations were, in general, worse, being TF the one with closest results to TF-IDF (as expected). Moreover, with the D2V representation it was necessary to reduce the dimensions of the vocabulary used (to 460 features) in order to be able to run the models in a reasonable amount of time and without exceeding the RAM limit.

In the sub-problem `L1`, both LP and LS overcome (with scores of 0.8077 and 0.8462) and the baseline model but Self Learning and EM algorithm obtain quite bad results, having the latter accuracy and F_1-scores lower than 50%. Plus, adding unlabeled data worsen the results for LP and LS.

As the results had lower than 90% accuracy/F_1-score, the dimensionality reduction techniques were applied. The number of components used for PCA and LSA was 57, 931 and 460 number of components for 0%, 10% and 30% of unlabeled data, respectively, for LDA it is always used 95 and for Chi2 and MI 931 components. PCA and LSA had very similar results, as well as Chi2 and MI. LDA had worse results, besides being very time consuming. In general, the results did not improve from the base cases and the few that improved had a difference from less than 2%.

The other text representations did not have better results, with the exception of the TF, were the

LP had an improvement of more than 6% but the LS had much worse results.

For the sub-problem `L2` the results were quite bad in general and even the baseline model, LP and LS had results below 56%. The scenario 2 was then tested.

Using K-means with 65 clusters (11 of them were repeated but only 9 had more than 4 labels associated and for those, the maximum number of labels associated was 6), the results were the same for both LP and LS, having accuracy$=$ 0.9766 and F_1-score$=$ 0.9742 without unlabeled data and accuracy$=$ 0.9649 and F_1-score$=$ 0.9610 (or 0.9602) with 10% or 30% of it.

On the other hand, using DBSCAN with `eps`$=$ 0.65 and `min_samples`$=$ 2 (which corresponds to 42 clusters with a maximum length of 10 labels and including the noisy cluster '-1', where the labels with only 1 sample belong to), the results were equal for LP and LS, being accuracy$=$ 0.9762 (regardless the amount of unlabeled data) and F_1-score$=$ 0.9722 with no unlabeled data and F_1-score$=$ 0.9704 with 10% or 30% of it. Therefore, DBSCAN gets a better F_1-score (although the accuracy is slightly worse, the difference is less than 0.05%) with a lower number of clusters, however, K-means has a lower maximum cluster length.

As the unlabeled data seemed not to have a significant influence in the results, the scenario 3 was also tested and the score obtained was 0.9157.

In the sub-problem `L3` all models had results below 50% accuracy/F_1-score, so the process followed was the same as in the previous sub-problem.

The best set of parameters found for K-means was to use 50 clusters (with maximum length of 8 labels) and for DBSCAN was `eps`$=$ 0.5 and `min_samples`$=$ 2 (which corresponds to 49 clusters, with the noisy cluster included, and a maximum cluster length of 8 labels). The results were again quite good, being slightly better with the K-means, where LP had always the same accuracy/F_1-score, 0.9935, regardless the amount of unlabeled data, and the LS had always accuracy$=$ 0.9870 and then F_1-score improving from 0.9810 to 0.9870 with the 10% or 30% of the unlabeled data.

Again, the scenario 3 was also tested and the score obtained was 0.9139.

Although the results from the sub-problem `CAUSE` are quite better than the sub-problem `L3` (except for EM algorithm and LS), the accuracy/F_1-score were still below the 50% for all the models and so the approach from scenario 2 was also applied, noticing that this time the results were not as good as in the last two sub-problems.

With the K-means algorithm, the number of clusters used was 65 (with maximum cluster length of 11 labels), where LP obtained accuracy$=$ 0.9788 and F_1-score$=$ 0.9779 and LS had much worse results, with accuracy$=$ 0.7196 and F_1-score$=$ 0.6509, both LP and LS having worse results with the increase of the amount of unlabeled data.

Using DBSCAN, with `eps`$=$ 0.65 and `min_samples`$=$ 2 (which corresponds to 62 clusters, including the noisy cluster, with maximum length of 12 labels), the performance was similar but with a significant improvement in the LS results, with accuracy$=$ 0.8389 and F_1-score$=$ 0.8001, and slightly worse results for the LP, with accuracy$=$ 0.9722 and F_1-score$=$ 0.9591.

It followed the scenario 3 approach, with a score of 0.7460, which corresponds to a difference of more than 20% when compared to the best accuracy/F_1-score results obtained with the approach from scenario 2 and yet being better than the results of LS (regardless the amount of unlabeled data used).

When assessing the `L0` labels associated with the labels in each of the other sub-problems, it was observed that the sub-problem `CAUSE` was the only one that had labels with more than one `L0` labels related to them, which could explain this huge difference in the results, when compared to the sub-problems `L2` and `L3`. Moreover, this was also the sub-problem with the highest number of features, since the set of all the combinations of the previous sub-problem's labels is being included in the vocabulary. Another interesting characteristic of this sub-problem's data is that it has unique observations associated to each of the 3 `L0` labels, whereas the other sub-problems only have unique samples associated with one of them, 'TV' (although in `L2` and `L3` one of the unique observations corresponds to other `L0` label, 'Voz').

5.2. Discussion

In general, the EM algorithm was the model with the worst performance, although it is very fast, as Self Learning. This is a bit unfair, since the parameter α is intrinsically defined and therefore is always the same, which could not correspond to the best value. The Self Learning algorithm has better results than the EM algorithm, but usually worse than the rest of the models. Sometimes it has the same results as the baseline model, which is normal since it is based on the NB classifier and it is usually not affected by the unlabeled data (only in sub-problem `L1`).

It is difficult to define which model is better between LP and LS, since they do not always use the same kernel function. What could be observed is that the 'knn' kernel was typically associated with worse results, except in the sub-problem `L0`. Moreover, although it was expected that 'knn' would require less iterations and would be faster than 'rbf', the latter was not true in he majority of the cases, which was most likely related with the number of neighbors chosen and the amount of observations used being extremely high. Note that LS tries to correct some possible mistakes in the original labels by updating the labels of the known nodes to be consistent with its neighbors, while performing label propagation. This task is related to the α parameter and could also increase the computation time.

The dimensionality reduction techniques seemed not to have significant influence on the results and therefore were not further explored. The TF-IDF appeared to be the most appropriate text representation to use, due to its better performance in terms of results but also since it was the fastest method.

Contrary to what was expected, the use of the unlabeled data had, in the most of the experiences, no effect or even a negative effect in the results. This could be due to the class imbalance (present in all sub-problems) but also due to the fact that the data does not satisfy the SSL assumptions. Although the class imbalance issue could not be successfully solved, since there was a significant number of unique observations that shouldn't be removed (as a big portion of information would be lost and the results were actually worse), some of the SSL assumptions were forced to be satisfied in scenarios 2 and 3.

In the sub-problems `L2`, `L3` and `CAUSE` it is clearly better to use the scenario 2 or 3 approaches than the scenario 1. In fact, for the sub-problem `CAUSE`, the best approach is the one from scenario 2, when using the LP.

Although the approach from scenario 3 was much faster, it was ignoring the majority of available data (the unlabeled set) and so the scenario 2 should be the implemented approach, despite the apparent irrelevance of the unlabeled data (caused by the class imbalance).

The results, in general, become worse as one goes further in the sub-problems, which could be related with the increase in the number of labels associated to them (that also increases the computation time) but also with the number of unique observations. Moreover, the dependence between the sub-problems is only represented by the words added to the texts, corresponding to the labels of the previous sub-problems, which might not be sufficient.

6. Conclusions

6.1. System Limitations and Conclusions

The main goal of this thesis was to apply text mining and text categorization techniques to a set of troubleshoot guides from a telecommunications call center and use it in order to improve the recommendation system performed by its technical assistants. Since the data provided was mainly unlabeled, with a few labeled observations, the idea was to explore SSL models and how they could make use of all the available data.

This work was associated with several limitations, starting with the fact that the data was unstructured and most of the data observations were unlabeled. Moreover, as a lot of variables were involved in the process of diagnosis and troubleshooting of the client's problem, the analysis and the model construction had to be divided into 5 different sub-problems. In all of them, there was an evident class imbalance within the labeled set. Plus, since the text data was extracted by a web crawler that only fetched the web pages text elements (e.g. was not able to extract text included in images), some parts were missing and therefore the quality of some texts was compromised.

Although the results for the first two sub-problems were good, the results for the following sub-problems in scenario 1 were quite bad. This way, it was not possible to achieve the full initial objective of constructing a new database, based on those results, that could be added to the recommendation system, indicating the correct troubleshoot guide to follow after all diagnosis steps were complete. Therefore, the alternative approach was to categorize the texts into a set of labels, smaller than the entire set, for each of the five sub-problems, starting by clustering the original set of labels and then classify the documents into the resulting clusters. The idea was to then provide these results to a specialist in order to help him with the diagnosis variables and troubleshoot guide association and thus saving time. This alternative solution was successfully achieved, obtaining more than 92% accuracy/F_1-score for the 3 sub-problems in which it was applied, however, the effect of using it did not seem to be significant as it would be expected. This is related with the similarity of topics contained in each text, which makes the SSL assumptions difficult to satisfy, even if clustering the possible classes, as there is still the class imbalance problem that affects the quality of the clustering and the class separability (e.g. the same label being included in several clusters).

6.2. Future Work

Having the partial solution for the main goal of this work, there are still some possible developments that could be explored and implemented in order to enhance and complete this solution.

The first thing to do would be to implement the idea of attributing the importance of each label (based on the TF-IDF mechanism) in each cluster, suggesting the labels from the most to the least probable one and giving an extra help to the technical specialist that would do the association between the diagnosis variables and the troubleshoot guides.

Another important development would be to improve the quality of the texts used to train these model, by finding or developing a better web crawling method, that could extract text from images, but also by exploring the text preprocessing techniques for the Portuguese language and multilingual problems (since the texts are mostly written in Portuguese but have also some English expressions).

It could also be interesting to explore other SSL algorithms and implement them, since the solutions from Python's `sklearn` package (LP and LS) are very time consuming and need a lot of computer RAM when dealing with huge amounts of samples and, moreover, do not manage well with class imbalance.

After improving the data quality and the solution presentation, a specialist should use this in order to build the desired dataset to be added to the existing recommendation system and hence complete the solution, making the troubleshoot process of the call center more efficient and, in that way, improving it.

References

[1] J. E. Alvarez and H. Bast. A review of word embedding and document similarity algorithms applied to academic text. 2017.

[2] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4), 2009.

[3] D. Berrar. Bayes' theorem and naive bayes classifier. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics; Elsevier Science Publisher: Amsterdam, The Netherlands*, 2018.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 2003.

[5] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1977.

[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 1996.

[8] J. Jauregui. Principal component analysis with linear algebra. *Philadelphia: Penn Arts & Sciences*, 2012.

[9] D. Jurafsky and J. H. Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2020.

[10] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown. Text classification algorithms: A survey. 2019.

[11] C.-H. Lee and H.-C. Yang. Construction of supervised and unsupervised learning systems for multilingual text categorization. 2009.

[12] D. D. Lewis. Evaluating and optimizing autonomous text classification systems. 1995.

[13] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2), 2003.

[14] Z. Liu. *High performance latent dirichlet allocation for text mining*. PhD thesis, Brunel University School of Engineering and Design, 2013.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. 2013.

[17] B. Rosario. Latent semantic indexing: An overview. *Techn. rep. INFOSYS*, 240, 2000.

[18] M. Toman, R. Tesar, and K. Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, 4, 2006.

[19] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2), 2015.

[20] J. R. Vergara and P. A. Estévez. A review of feature selection methods based on mutual information. 2014.

[21] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. 1997.