Automating the Response Processes in TAP PORTUGAL's Social Networks

Tiago Simões
tiagosimoes@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

May 2017

Abstract

The rapid dissemination of information and ideas, in combination with the rise of the development of instant communication, lead to an accelerated growth of short texts. Since short texts might enclose valuable intelligence, mining these sources has become of increased interest for corporations. It was in this context that TAP PORTUGAL showed interest in the elaboration of a study, based on machine learning algorithms, to identify the comments that its clients make on Facebook and Twitter in 4 typologies: Praise, Complaints, Questions and Suggestions. Analyzing the new created corpus, it was possible to characterize it as: sparse and short; multilingual; unlabeled; noisy; imbalanced; and non-standard. From the annotated corpus and acquired knowledge of manually labeling it, we retrieved and analyzed predominant complaints, praises, questions and suggestions. The conducted experiments using machine learning algorithms pinpointed k-Nearest Neighbors and Support Vector Machine as the best classifiers among the ones studied, achieving very high scores in certain conditions. Despite the good scores achieved by both classifiers, Support Vector Machine was clearly the most robust model presenting very good results when reducing the training data as well.
Keywords: Text Categorization, Social media, TAP PORTUGAL

1. Introduction

Automated text categorization is the process that consists of the automatic classification of documents into a fixed number of predefined categories, and it has become one of the core techniques for handling and organizing text data. Falling at the crossroads of Information Retrieval (IR) and Machine Learning (ML), automated text categorization allows to identify relevant information, having a significant impact in applications such as content management, contextual search, opinion mining, product review analysis and spam filtering [1].

TAP PORTUGAL (TAP) has an active participation on Customer Care service on its social networks, and it has an application to manage the response task of comments, messages, posts and tweets from its social media services. However, it is desired to have an application to automatically categorize all the feedback, that otherwise would have to be manually addressed and sorted by the Contact Center operators.

Structured and categorized feedback allows a better understanding of clients' needs. One of the main reasons for applying data mining methods to text documents collections is to structure them [2].

In this document we explain the process of: (i) retrieve comments, messages, posts and tweets from TAP's clients on TAP's social media channels; (ii) create a corpus from the gathered text; (iii) create an annotated dataset with the following categories: complaints, praises, questions and suggestions, that can be used to train supervised ML algorithms; (iv) study and evaluate ML algorithms able to classify automatically the corpus in the given categories.

The document is organized as follows: Section 2 introduces essential concepts to understand the problem and the solution proposed, Section 3 details the process of creating the corpus, along with a detailed description of it. Section 4 explains the setup of the experiments using ML classifiers and respective results, and also the knowledge retrieved from the built corpus. Section 5 closes the document summarizing its core aspects, and proposing directions for future work.

2. Fundamental Concepts and Related Work

2.1. Text Representation

In the context of Natural Language Processing (NLP) and IR, the most common approach is to represent text is through unigrams and its respective counts, also known as, Bag of Words (BoW) [3, 4, 5, 6]. In this representation all the information about the order of the words in a docu-

ment is lost and the text is transformed into a bag of words together with word frequency counts. Longer n-grams have been considered in previous text classification works. Although, they failed to provide consistent results, depending deeply on the problem and dataset [3, 4]. In the context of text categorization, the BoW representation is usually coupled with a weighting scheme, such as Term Frequency-Inverse Document Frequency (TFIDF) and binary weighting (presence/absence or 1/0) [7].

Defining a set of documents as $D =< d_1, \ldots, d_n >$, TFIDF determines the most important terms in a document d.

It is widely recognized that TFIDF is one of the most popular term weighting schemes in text categorization [8, 9, 10, 4]. Its formula can be given by:

$$\text{tfidf}(w) = \text{tf}(w) \times \text{idf}(w) \tag{1}$$

$$\text{idf}(w) = \log\left(\frac{N}{N_i}\right) \tag{2}$$

Where w is the term, tf is the term frequency in a given document, N is the total number of documents the dataset contains, N_i is the number of documents in the dataset that contains the term w. The tfidf value is proportional to the number of times a term w appears in the document, but is offset by the frequency of the term w in the document, which accounts for the fact that some terms appear more frequently [5, 11].

2.2. Short Text Categorization

Extensive research has been made to text categorization, however, not so much for short text categorization [12]. Due to the sudden growth of e-commerce and online communications, short texts have become the most common type of text found on the web [12]. These texts are growing rapidly and they can be encountered in many application areas, such as online chats, Short Message Service (SMS), tweets, blog comments and short news. Unlike conventional documents (e.g. reports, magazines), these are usually: (a) noisier - abundance of irrelevant content; (b) less structured - misspells and non-standard terms are often found; (c) noticeable shorter - no longer than a few sentences; (d) large scale - they can be found in every application, process, messing system, among others; (e) unlabeled - majority of texts are not identified as belonging to a category.

Researches have demonstrated that supervised ML algorithms have been applied to an appreciable number of cases with different benchmark collections and have achieved satisfactory results in both short-text classification and text classification [5, 4, 8, 13, 1, 12, 14, 15].

k-Nearest Neighbors (kNN) is a well-documented example-based classifier, that determines the class of an object according to its closest k neighbors [16]. It is known to have a fast training phase and be one of the simplest and best performing text classifiers [7].

In the training phase, kNN stores the representations of all training documents and the respective class labels, instead of building explicitly "declarative representations of categories" [17]. On the classification phase, kNN computes the distance between the new instance x and every training example x_i, assigning to x the class of its k nearest neighbors.

Taking in consideration that most of real word datasets are unbalanced establishing the number of neighbors k is of utmost importance [7, 18]. When $k = 1$, the new document is simply attributed to its nearest neighbors class, while high k values might lead kNN to attribute the new document to the majority classes.

Another characteristic of kNN is that it tends to incorrectly classify a document holding features never seen before (i.e., in the training phase), because it fails to calculate the distance value between the new document and the existing documents (undefined distance).

The main drawbacks of kNN are the "high computation cost of classification, because for each test document it must compute its similarity to all of the training documents" and the required space to store all training examples [7, 17].

Multinomial Nave Bayes (MNB) is a popular probabilistic model designed for text classification based on Naive Bayes (NB) theorem, known to be computational efficiency and a relatively good predictive performance. [19, 4, 20]. This classifier calculates the maximum probability of a document belonging to a category [19]. The probability of a document d belonging to a class c is given by Eq. 3.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{3}$$

Where $P(t_k|c)$ is the conditional probability of a term t_k occurring in a document of class c; n_d is the total number of terms in document d and $P(c)$ is the prior probability of a document occurring in class c [21].

MNB assumes that a document is defined by the BoW representation, meaning that what is relevant for this algorithm is the number of occurrences of each term, and not the order of appearance in a document; and that the probability of different features/terms are independent giving the class.

Similarly to kNN, this classifier tends to misclassify documents holding new features never seen

before (i.e., in the training phase), because it fails to calculate the probability of the new features in the existing documents (zero probability).

Support Vector Machine (SVM) is a binary ML algorithm commonly used for text classification problems that aims to find the optimal hyperplane that separates two classes [22, 13, 5, 8, 4].

A good of separation is achieved by the hyperplane that has the largest width of a band around a decision boundary without any training samples, also known as, functional margin.

Unlike other classifiers, SVM might not use all documents provided in the training phase, i.e., the margins are built only by documents near to the classification border; it can build an irregular border to separate positive and negative training documents; "not all the features (unique words) from the training documents are necessary for classification" [23, 7].

SVMs can efficiently perform linear and non-linear classifications using what is called the *kernel trick*, implicitly mapping their inputs into high-dimensional feature spaces, i.e., it converts not separable problem into a separable problem [7].

This classifier is characterized by being fast in the testing phase and highly accurate even in multi class problems.

The major drawbacks are the long training phase (this classifier is highly dependent on the number of training examples and number of features); and the difficulty of picking an adequate kernel.

Fuzzy Fingerprints (FFP) is a machine learning algorithm originally used to text authorship that has an analogy with human fingerprints [8, 24]. Each human has its own fingerprints, so does a set of texts from the same author. As the author state: "The main concept behind this algorithm is that authors have a stable enough behavior that allows a set of features to be extracted, fuzzified and then compared." [24], i.e., the most relevant features, such as words and punctuation, of each author are gathered from a set of texts, and through a fuzzifing function, a unique fingerprint is generated. In order to detect the authorship of a text, a fingerprint is created based on text's features and then compared to authors' fingerprints employing a similarity measure. Despite being originally used for text authorship, the approach followed by this classifier proved to be effective and efficient in topic detection as well.

Rosa et al. [8] proposed an adaptation of the original classification method that performs better in short texts, like tweets, when working with a high number of classes.

Due to the constrain of 140 characters in tweets, it was unreasonable to count the number of individual word occurrences. Therefore, "its features should be as unique as possible in order to make the fingerprints distinguishable amongst the various topics". To do that the author applied an adaptation of the popular Inverse Document Frequency (idf): Inverse Topic Frequency (itf), "where topics are used instead of documents to distinguish the occurrence of common words" amongst topics [25].

$$\mathrm{itf_v} = \log\left(\frac{J}{j_v}\right) \tag{4}$$

In Eq. 4, J is the total number of topics and j_v stands for the number of #topics where the words v is present.

Due to the small size of a single tweet, a new scoring metric had to be developed, Tweet-Topic Similarity Score (T2S2).

$$\mathrm{T2S2}(\Phi, T) = \frac{\sum_{\upsilon} \mu_{\Phi}(\upsilon) : \upsilon \in (\Phi \cap T)}{\sum_{i=0}^{j} \mu_{\Phi}(w_i)} \tag{5}$$

T2S2 approach "tests how much a tweet fits to a given topic", taking into account the size of the tweet. It returns values between 0 and 1, where the lowest score indicate that the tweet in questions had no similarity to the topic fingerprint. As the authors enunciates "T2S2 divides the sum of the membership values $\mu_{\Phi}(\upsilon)$ of every word υ that is common between the tweet and the #topic fingerprint, by the sum of the top j membership values in $\mu_{\Phi}(w_i)$, $w \in \Phi$.".

The area of text categorization is vast and it is impossible to cover all the different algorithms in detail [26]. Therefore we covered the most relevant for this work.

It is important to note that Neural Networks have proved to be an effective classifier in this area. However these approaches usually require enormous quantity of data and time to train. Given the available data we discarded neural networks approaches.

2.3. Evaluation Metrics

To assess the classification of the ML algorithms we chose the well-know metrics Recall (Eq. 7), Precision (Eq. 6), Accuracy (Eq. 9) and F-Measure (Eq. 8) [27, 12].

Before presenting the formulas of the evaluation metrics, it is essential to explain the statistics concepts belonging to them [28, 12]:

True Positive (TP) When an instance is correctly classified as true.

False Positive (FP) When an instance is incorrectly classified as true.

True Negative (TN) When an instance is correctly classified as false.

False Negative (FN) When an instance is incorrectly classified as false.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F - Measure = 2 \times \frac{P \times R}{P + R} \tag{8}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

2.4. Related Work

An interesting survey [12] covering numerous short text classification approaches was released recently. This survey overviews numerous methods for short text classification, tackling traditional ML classifiers like kNN, NB, Maximum Entropy and SVM. The authors argue that these classic methods tend to underperform in comparison with: (i) Short Text Classification Based on Semantic Analysis; (ii) Semi-Supervised Short Text Classification; (iii) Ensemble Short Text Classification; and (iv) Real-time Classification of Large Scale Short Text. Despite a considerable number of studies suggesting that some of these classifiers perform very well in some datasets, many of these technologies are in the initial stage and some problems were found in the area of "dynamic short text stream, multi-label short-text classification, comment emotional classification spam filtering, and topic tracking control.". Since, this survey covers well most of the state of the art techniques for short text classification, it might seem reasonable to referencing it and redirect the reader to it.

3. Corpus

3.1. Data Sampling and Analysis

Several sets of data were retrieved from TAP's Facebook and Twitter accounts, each of them combining comments, posts and messages from both TAPs customers and TAPs contact center operators. Using NEXT Analytics Dashboard Refresh Tool [1] and SONAR [2], it was possible to extract samples of public and private data flowing through these two social media platforms. In total, it was gathered more than 15 000 uncategorized and unstructured comments, messages, posts and tweets from social platforms and then merged to create an appreciable corpus.

Language	Portuguese	English	Other
Distribution	70%	25%	5%

Table 1: Language distribution across the dataset.

Source	Social Media Users	TAP PORTUGAL Contact Center
Distribution	71%	29%

Table 2: Source distribution across the dataset.

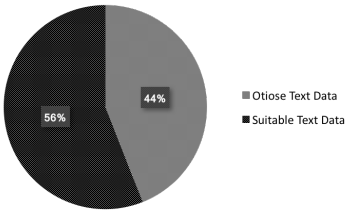

Figure 1: Distribution of usefulness of the original corpus.

Texts considered irrelevant to achieve the goal of this work, such as, comments from TAP Contact Center (Table 2) and messages not in Portuguese (Table 1) were identified and then removed.

As we can see from Figure 1, a rather small corpus (approximately 8400 texts) remained to train the classifiers.

3.2. Corpus Characteristics and Preprocessing

Text found in social networks' conversations and discussions tends to be different from text in books and articles for several reasons. Analyzing the corpus, it was possible to characterize it as: (a) sparse and short - users tend to avoid type long messages, as the average text in the corpus has approximately 24 terms [3]; (b) non-standard - users do not to pose as much care into an accurate choice of words or in the structures of their sentences as they, probably, otherwise would, in other contexts. That may lead to different types of ambiguities, such as lexical, syntactic, and semantic [29]. Therefore, analyzing and extracting information patterns from such data sets is a complex task; (c) noisy and imbalanced - as it can be seen in Figure 2, most of the texts retrieved belong to a single class, which contains the irrelevant messages, i.e., the noisy messages; (d) multilingual corpus - 30% of the corpus was in English, French, Italian, German, Spanish, among others; (e) unlabeled;

Another particularity in social media texts are emoji. They add meaning and manifest emotions, enriching the text. E.g. ":D" and ":(".

It should, however, be noted that due to the limitation of 140 characters in tweets, acronyms and abbreviations are commonly used in social networks [5]. E.g. "b4" is short for "before". In some studies it is used a customized stop-words list that

[1] Developed by Next Analytics.

[2] Developed by Whale.

[3] A term comprehend anything from a number to punctuation or a word.

identify and transforms acronyms in real words. Although we are dealing with social networks text and this is a common practice in this context, its presence is estimated to be under 4% in all corpus. Hence, it was decided to ignore it.

The baseline techniques adopted to normalize the corpus were: removing Uniform Resource Locator (URL)s, email addresses, Twitters usernames, diacritics and capitalization; parsing HyperText Markup Language (HTML) entities and standardizing emoji and words; tokenization.

3.3. Categories

TAP defined as main categories: Complaints, Praises, Questions and Suggestions. However, whilst the annotation work proceeded it was possible to perceive that some text suited better other categories. As a result, we added the categories: Acknowledgments, Insults and Other.

Complaint When a client states that a service they experienced was unsatisfactory or unacceptable. E.g. "It is unacceptable that the cabin crew does not speak English.".

Praise This label is applied when a customer writes a positive remark that emphasizes a service or the company quality. E.g. "It is a pleasure to fly with TAP. It is my 30th flight with you.".

Question It is given a "question" label to a text that demands information related to any service provided by the company. E.g. "Can I use my miles to buy a flight from Lisbon, Portugal to Oporto, Portugal?".

Suggestion Sometimes people want to active participate in a companys future. An idea that can improve a service provided by the company or the creation of a new service it is labeled as "suggestion". E.g. "TAP should consider providing XL seats.".

Acknowledgment A quick look to the dataset allowed us to see that there were many messages acknowledging the information given by the contact center. As consequence, this category was created. E.g. "Thank you for the information.".

Insult When someone express hatred to the company without further explanation. E.g. "Worst airline ever!".

Other Every text that is in a language other than Portuguese or is unsuitable to any of the categories listed above, it is classified with this label. Since TAP's social platforms are public, everyone can potentially publish a text of any nature or purpose (i.e. noise). E.g. "Last time I flew on a TAP's plane was in 1999.".

3.4. Validation

The categorization of the built corpus was conducted by a single annotator. Therefore it was necessary to validate its quality. In order to assess the accuracy of the instructions to categorize a text and to measure the reliability of the categorization process, it was asked to two students of Instituto Superior Técnico (IST) to manually classify 104 instances of the dataset collected. It was obtained a Krippendorffs α of 0.664, i.e., there were some inconsistencies, but overall a considerable amount of the classification was identical.

Inspecting carefully the classification of the annotators, the prominent challenge was to classify ironic comments, i.e., complaints masked as praises, and suggestions that can be classified as questions and vice versa.

3.5. Distribution of the Dataset

The chart displayed in Figure 2 plots the distribution of the built corpus over the 7 classes. These divergent results suggest that we are facing an unbalanced dataset, where the category that owns gibberish text is clearly the majority class.

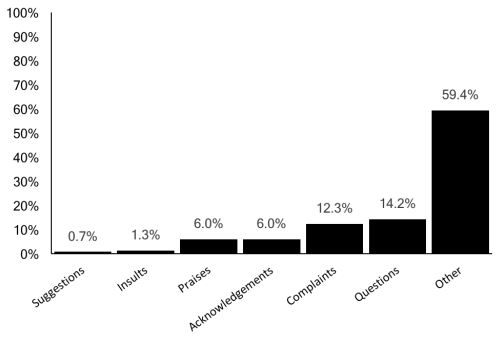

Figure 2: Distribution of categories across the dataset.

It is also important to notice that only 33.3% of the suitable text fits into categories established by the company. Although, if we consider the entire corpus, only 22.7% is relevant to the company - this proportion comes closer to the Pareto principle (80/20 rule).

4. Experiments and Results

4.1. Experimental Setup

Python 2.7 was the selected programming language to develop preprocessing scripts and FFP adapted to short texts. WEKA 3.6.0 was the main tool used to evaluate kNN, MNB and SVM performance. Both offer simple and efficient tools for data mining (including text classification), supported and used by academic communities worldwide.

4.1.1 Preprocessing

In order to maximize the classification results of the prediction models and to complement the normalization techniques already mentioned, more

techniques were considered whilst developing this work, such as (i) removal of stop-words; (ii) removal of punctuation and (iii) substitution or removal of numbers.

It is known from the literature that BoW is the most used text representation and TFIDF the most used weighting scheme in text classification, making the most unique features of each document stand out. Therefore, this is the default format of text data representation selected for this work, except for FFP. This uses a variation of TFIDF - Term Frequency-Inverse Topic Frequency (TFITF).

Information Gain (IG) and Chi-Square are the most effective feature selection methods in text classification [27]. However, the former shows slightly better results in short texts categorization problems [30]. Hence, it was the only feature selection method considered in the experiments.

In order to select the IG threshold, we conducted several experiments with different values. We empirically selected 0.001 as the minimum value for a feature to be selected.

4.1.2 Evaluation

The experiments using MNB, SVM and kNN classifiers were executed in WEKA. We split the dataset in 90% for train and validation (for validation purposes we used stratified 10 fold cross validation) and 10% for testing. On the other hand, FFP adapted to short texts used 90% of the data for training the model and 10% to test. No cross validation was performed. Both training and testing files were shared amongst experiments, making them comparable.

To assess the classification of the ML algorithms we chose the well-know metrics Recall (Eq.7), Precision (Eq.6), Accuracy (Eq.9) and F-Measure (Eq.8) [27, 12].

4.2. Experimental Results and Analysis

4.2.1 k Nearest Neighbors

To begin with, it was important to understand the number of neighbors to use. Since we were dealing with an unbalanced dataset, using a high k might have lead the classifier to assign the label "Other" to all instances. Table 3 displays the obtained results of kNN, considering baseline preprocessing techniques and different values k.

The results displayed in Table 3 are in exceptionally good agreement with previous studies, that supported kNN as a great predictor. Assigning a text to its nearest neighbor ($k = 1$) achieved very good results but probably due to overfitting.

Higher values of k failed to achieve good results due to the unbalanced nature of the dataset. Increasing the number of neighbors led kNN to classify more instances as "Other", since this was the majority class and it held more terms than any other class (See Table 4). Therefore, considering $k > 1$, kNN tended to find more similarities with this class than the others, being harder to correctly classify instances belonging to the other classes.

kNN	Words To Keep	FS	Wei. Avg Recall	Wei. Avg. Precision	Wei. Avg. F1	Acc
1NN	1000	-	**88.70%**	**89.90%**	**88.90%**	**88.68%**
		IG	**88.70%**	**89.90%**	**88.90%**	**88.68%**
	10000	-	88.40%	89.60%	88.60%	88.35%
		IG	**88.70%**	**89.90%**	**89.90%**	**88.68%**
3NN	1000	-	77.50%	77.40%	73.20%	77.49%
		IG	76.10%	75.20%	71.60%	76.14%
	10000	-	76.10%	77.30%	72.70%	77.04%
		IG	77.70%	77.30%	72.70%	77.04%
5NN	1000	-	75.80%	77.10%	70.40%	75.81%
		IG	75.90%	76.00%	70.80%	75.92%
	10000	-	75.40%	77.40%	69.80%	75.36%
		IG	76.70%	78.20%	71.70%	76.70%

Table 3: kNN performance employing baseline preprocessing.

A	E	I	O	P	S	R	<– classified as
10	0	0	3	0	0	0	A (acknowledgments)
4	46	1	62	0	0	0	E (praises)
0	1	3	1	0	0	0	I (insults)
5	6	1	599	1	0	0	O (others)
3	2	0	40	11	1	0	P (questions)
0	1	0	14	0	0	0	S (suggestions)
3	3	2	60	2	0	8	R (complaints)
A	E	I	O	P	S	R	<– classified as
13	0	0	0	0	0	0	A (acknowledgments)
0	94	0	9	0	0	10	E (praises)
0	0	2	2	0	0	1	I (insults)
1	14	0	567	4	2	24	O (others)
0	1	0	5	40	4	7	P (questions)
0	4	0	2	0	7	2	S (suggestions)
0	0	0	8	0	1	69	R (complaints)

Table 4: Confusion matrices for kNN considering the 1000 most relevant words employing baseline preprocessing. On the top, $k = 5$. On the bottom, $k = 1$.

Further conclusions might be taken from Table 3 about kNN's behavior in this dataset. It can be reasoned out that employing IG and/or gathering more features per class, barely had impact on the classifier's performance. Because kNN considered only the most relevant features to identify a class rather than using them all, both practices had almost no effect.

Despite other values of k and $WordsToKeep$ presenting good results, they proved to be worse than $k = 1$ and $WordsToKeep = 1000$. Due to the good results summarized in Table 3 with these parameters, the experiments realized focused on $k = 1$ and $WordsToKeep = 1000$. With these settings we kept our model small and efficient. Larger values would only have made the model larger and slower. IG was also excluded because it proved to be irrelevant to kNN in this dataset.

Removing all Natural Language Toolkit (NLTK)[31] Portuguese stop-words, numbers and all the punctuation (except "!" and "?") kNN

accomplished very high scores - $F1 = 89.10\%$, $R = 88.90\%$, $P = 90.10\%$ and $Acc = 88.85\%$.

kNN with $k = 1$ in general implies overfitting. As we can see from the confusion matrices (Table 5), as we trained the classifier with more data, it overfitted to all categories enhancing the quality of the prediction model.

A	E	I	O	P	S	R	<– classified as
145	2	0	15	3	0	0	A (acknowledgments)
6	87	3	69	4	0	4	E (praises)
0	6	14	16	1	0	1	I (insults)
22	27	13	1495	22	0	18	O (others)
18	5	4	188	156	2	12	P (questions)
1	4	1	8	4	6	0	S (suggestions)
7	18	3	152	40	2	139	R (complaints)

A	E	I	O	P	S	R	<– classified as
13	0	0	0	0	0	0	A (acknowledgments)
0	94	0	9	0	0	10	E (praises)
0	0	2	2	0	0	1	I (insults)
1	14	0	567	4	2	24	O (others)
0	1	0	5	40	4	7	P (questions)
0	4	0	2	0	7	2	S (suggestions)
0	0	0	8	0	1	69	R (complaints)

Table 5: Confusion matrices for 1NN considering the 1000 most relevant words employing baseline preprocessing. On the left, split 67% / 33%. $F1 = 72.20\%, Acc = 74.44\%$. On the right, split 90% / 10%. $F1 = 88.90\%, Acc = 88.68\%$.

4.2.2 Multinomial Naïve Bayes

From Table 6 it can be seen that: (i) as expected, MNB increased performance as the number of features rose. Due to its nature, this classifier tends to perform better when it is trained with large amounts of data rich in features [19, 32, 21]; (ii) as the number of features increased we could have spared the usage of the feature selection tool. This was cause by the IG configuration used (only selects features with scores higher than 0.001), that led to a loss of features, which were crucial for MNB correctly classify instances; and finally (iii) normalizing all data to the average length of training instances appeared to have a positive impact in the performance of this classifier when $WordsToKeep > 3000$ and IG was discarded. Because MNB assumes that features are not intertwined, long texts might negatively effect parameter estimates due to their higher terms frequencies [21]. Normalizing the length of a text eliminates all information on the length of the original text, avoiding this problem.

In spite of MNB performing better with large amounts of features, our results suggest that some features, such as punctuation marks and stop-words might be, as expected, disadvantageous for this classifier.

The best result using MNB was obtained considering 10000 words when data was normalized to the average of the length of training corpus and in addition of the baseline preprocessing, the punctuation except "!" and "?" was removed ($R = 83.70\%, P = 87.20\%, F1 = 84.60\%, Acc = 83.74\%$). The increase of 1% in F1 and 1.10% in Accuracy derived mainly from the correct classification of "Others".

Norm.	Words To Keep	FS	Wei. Avg Recall	Wei. Avg. Precision	Wei. Avg. F-Measure	Acc
False	1000	-	76.50%	80.70%	78.00%	76.48%
		IG	76.60%	82.40%	78.50%	76.59%
	3000	-	80.00%	84.90%	81.20%	79.95%
		IG	79.50%	82.40%	80.50%	79.50%
	5000	-	81.20%	85.20%	82.20%	81.18%
		IG	80.00%	82.50%	80.80%	79.95%
	10000	-	**83.00%**	86.30%	**83.90%**	**82.97%**
		IG	80.00%	82.50%	80.80%	79.95%
True	1000	-	75.00%	82.20%	77.20%	75.02%
		IG	77.00%	82.10%	78.70%	77.04%
	3000	-	79.20%	85.40%	80.70%	79.17%
		IG	79.20%	82.50%	80.30%	79.17%
	5000	-	82.60%	**87.20%**	83.70%	82.64%
		IG	78.90%	82.20%	80.00%	78.94%
	10000	-	82.60%	86.50%	83.60%	82.64%
		IG	79.70%	82.60%	80.60%	79.73%

Table 6: MNB performance applying baseline preprocessing techniques.

4.2.3 Support Vector Machines

For assessing SVM's performance categorizing texts in "Acknowledgments", "Complaints", "Insults", "Praises", "Suggestions", "Questions" or "Other", it was considered the degrees, $d = \{1, 2\}$, and Polynomial and Radial Basis Function (RBF) kernels. These were available on a WEKA implementation based on Sequential Minimal Optimization (SMO) [33, 34, 35].

Table 7 presents the performance of Polynomial and RBF kernels applying the baseline methods described previously.

Ker.	d	Words To Keep	FS	Wei. Avg. Recall	Wei. Avg. Precision	Wei. Avg. F-Measure	Acc
RBF	1	5000	-	77.40%	78.50%	72.30%	77.37%
			IG	76.50%	77.50%	70.90%	76.48%
		10000	-	78.80%	78.80%	72.80%	77.82%
			IG	76.50%	77.50%	70.90%	76.48%
Poly	1	5000	-	**88.90%**	**89.90%**	**89.00%**	**88.91%**
			IG	87.00%	86.90%	86.50%	87.01%
		10000	-	87.60%	87.60%	87.20%	87.01%
			IG	87.00%	86.90%	86.50%	87.01%
	2	5000	-	88.10%	89.20%	88.30%	88.12%
			IG	88.10%	89.10%	88.00%	88.12%
		10000	-	88.20%	89.40%	88.40%	88.24%
			IG	88.10%	89.10%	88.00%	82.12%

Table 7: SVM Performance employing baseline preprocessing methods.

From the results displayed (Table 7), several conclusions can be extracted: (i) it was evident that Polynomial kernel outperformed RBF, scoring almost 20% higher in F1. Due to the nature of RBF kernel, this tends to overfit to the train dataset, making hard for the classifier to correctly classify text slightly different from what it was trained with; (ii) it was possible to compare the influence of applying IG on both kernels. Carefully inspecting the table, it was obvious that the best performances

were achieved when discarding IG. As Hotho et al. stated in [14], SVMs rarely required feature selection as they "inherently select data points (the support vectors) required for a good classification; (iii) Table 7 provided compelling evidence that the degrees $d = \{1, 2\}$, of the Polynomial kernel show similarly satisfactory results in NLP.

Due to low performance of RBF kernel and the difference in results manifested by IG, these were discarded earlier in the development of the project. An appreciable number of experiments, combining different preprocessing techniques were realized in order to determine the most effective preprocessing methods and fine tuning of SVM for this problem. However, the best obtained result ($F1 = 89.10\%$ and $Acc = 89.02\%$) was achieved using Polynomial kernel ($d = 1$), by just removing NLTK Portuguese stop-words in addition of applying baseline preprocessing techniques.

4.2.4 FFP adapted to short texts

It was adopted the same approach as Rosa et. al. in [8] (detailed in Section 2.2). However, instead of detecting the topic of a tweet, among dozens of possible topics, we are doing topic classification. Following the guidelines provided in that study, we replicated the code in Python, using auxiliary library such as NLTK [31], pandas [36], sklearn [37] and numpy [38]. For configuration purposes, the default values of $a = 0.2k$ and $b = 0.2$ were considered. These values proved to be the most effective in this classifier [24].

As it can be seen from Eq. 5, this algorithm's classification depends deeply from $T2S2$, which is the lowest threshold value for acceptance of a text belonging to a category.

Through extensive testing it was found that 0.1 might be the best value for $T2S2$ for this dataset. Hence, it was assumed this value as the lowest threshold value for acceptance of a text belonging to a category.

To evaluate FFP adapted to short texts performance in this dataset, two approaches were followed: (i) we created a fuzzy fingerprint for every category; (ii) we created a fuzzy fingerprint for every category, except for "Other". Since irrelevant texts were classified with this label, "Other" might not had a distinct fingerprint for itself. Therefore, all texts that failed to score a $T2S2$ above the defined threshold, were automatically classified as "Other", without being compared to others classes' fuzzy fingerprint. However, this approach demonstrated poorer results than the former. Despite "Other" being a sparse category, it was important to have a fuzzy fingerprint for itself.

Given the characteristics of short sized text, generating a fingerprint with a high k value resulted in large and sparse fuzzy fingerprints. Contrarily, a low k value generated rigid fuzzy fingerprints for each class. Therefore, the best results were based in the harmonious k value, 175, as this achieved the highest F-Measure in experiments (See Table 8).

k	Wei. Avg Recall	Wei. Avg. Precision	Wei. Avg. F-Measure	Accuracy
20	61.59%	69.05%	64.68%	61.59%
175	75.81%	**80.78%**	**77.49%**	**75.81%**
1000	70.43%	69.30%	68.37%	70.43%

Table 8: FFP adapted to short texts' performance employing baseline preprocessing methods.

After employing different preprocessing techniques, excluding just stop-words from corpus was the most effective preprocessing method - $R = 79.28\%$, $P = 79.90\%$, $F1 = 79.17\%$, $Acc = 79.28\%$.

4.2.5 Classifiers Performance Overview

One of the goals of the thesis was to study the behavior of some ML classifiers, presenting a viable solution for TAP. All the classifiers yielded good results. However, kNN and SVM stood out with very good performances, making these the best choices for the given dataset.

Rosa et. al [8] proposed an adaptation of Fuzzy Fingerprints for topic detection for on-the-fly processing of Big Data streams. In their study, FFP achieved outstanding results, outperforming kNN and SVM. However, this algorithm accomplished the lowest scores, which can be explained by the nature of our dataset; the difference in the problem; and the adaptation we made to deal with "Other" category. Despite obtaining worse results when comparing to kNN, MNB and SVM, this classifier training and testing time performance was several times better than any other.

MNB underperformance might be explained because: (i) kNN and SVM can be finer tuned; (ii) MNB might need more training instances.

Table 9 allow us to understand the impact of overfit in all classifiers. Clearly, SVM was the classifier that suffered less with this difference in the training set, and kNN the most. Supporting the hypothesis that kNN with $k = 1$ is proved to overfitting.

4.3. Dataset Retrieved Knowledge

4.3.1 Complaints

From the knowledge acquired while labeling the dataset and from its most common words, we can state that most complaints are related to: (i) long

Classifier	SVM		kNN		MNB		FFP		ZeroR	
Split	90%/10%	67%/33%	90%/10%	67%/33%	90%/10%	67%/10%	90%/10%	67%/33%	90%/10%	67%/33%
F1	**89.10%**	**82.90%**	**89.10%**	72.20%	84.60%	71.60%	79.17%	66.91%	44.30%	41.40%
Acc	**89.02%**	**83.52%**	88.85%	74.44%	83.74%	70.47%	79.28%	69.08%	59.39%	57.03%

Table 9: Overall best performances.

waiting times to get in touch with the contact center; (ii) flight delays; (iii) problems with luggage.

4.3.2 Praises

Most praises derived from sympathy of the cabin crew and the feeling of security when flying with TAP. Contrasting with data retrieved from complaints, clients tended to appreciate the all service provided by the company and avoiding specifying any kind of service.

4.3.3 Questions

Is the richest category this corpus has. From the knowledge acquired while labeling the dataset and from its most common words, we can state that most questions are related to: (a) miles; (b) booking; (c) passports; (d) connection flight between Lisbon and Oporto; (e) luggage; (f) difficulties using the website and (g) delays. Even though, most of the questions might be answered in TAP website, most clients go to social networks.

TAP website has a page dedicated to Frequently Asked Questions (FAQ)s, containing the answers to the majority of questions a costumer might have. However, FAQs webpage's content was not in conformity with the most frequently asked questions in TAP's Facebook and Twitter accounts. For example, no answer related to the keyword "passport" was found. For example a frequent question containing this keyword was: "I am Brazilian and I want to buy a plane ticket from Brazil to Portugal for next summer. However, I can not do it because the website is asking for a passport number which I do not have yet. Let me know how should I proceed."

4.3.4 Suggestions

Due to the limited number of instances collected, it was evident that "Suggestions" was the trickiest category to retrieve valuable knowledge from. However, words such as: "lisboa", "porto"and "taghournumber" stand out from the rest, supporting again the theory that this connection between these two cities has problems, and there is room for improvement.

5. Conclusions and Future Work

The conducted experiments pinpointed kNN and SVM as the best classifiers among the ones studied. Achieving F1 scores higher than 88% in certain conditions. Despite the good scores achieved by both classifiers, SVM is clearly the most robust model presenting very good results when reducing the training data as well. It was also possible to confirm that some preprocessing methods and supervised ML algorithms applied in text categorization might not be so effective in short text classification.

From the annotated corpus and acquired knowledge of labeling it, we retrieved and analyzed predominant complaints, praises, questions and suggestions.

The work results were presented to TAP on late March 2017. It was concluded that we successfully accomplished the objective, and due to the results, a classification model will be in production soon.

The next steps are defined as follows: (i) asking a professional to label more data in order to build a more reliable model; (ii) developing two more classification models: one for identifying the types of complaints and another to denote the most common suggestions; and (iii) studying sentiment analysis.

References

[1] M Ikonomakis, S Kotsiantis, and V Tampakas. Text Classification Using Machine Learning Techniques. *World Scientific and Engineering Academy Society Transactions on Computers*, 4, 2005.

[2] Gary Miner, John Elder, Thomas Hill, Robert Nisbet, Dursun Delen, and Andrew Fast. *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*. Academic Press, 2012.

[3] Fernando Batista and Ricardo Ribeiro. Sentiment analysis and topic classification based on binary maximum entropy classifiers. *Procesamiento del Lenguaje Natural*, 2013.

[4] Sida Wang and Christopher D Manning. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 2, 2012.

[5] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary. Twitter Trending Topic Classification. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Data Mining Workshops*, 2011.

[6] Ted Dunning. Statistical Identification of Language. Technical report, Computing Research Laboratory New Mexico State University, 1994.

[7] Fernando Batista and Joao Paulo Carvalho. Text based Classification of Companies in CrunchBase. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Fuzzy Systems*, 2015.

[8] H. Rosa, F. Batista, and J. P. Carvalho. Twitter Topic Fuzzy Fingerprints. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Fuzzy Systems*, 2014.

[9] Gerard Salton and Chris Buckley. Term Weighting Approaches in Automatic Text Retrieval. Technical report, Cornell University, 1987.

[10] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

[11] Joeran Beel, Stefan Langer, and Bela Gipp. TF-IDuF: A Novel Term-Weighting Sheme for User Modeling based on Users' Personal Document Collections. In *Proceedings of the iConference*, 2017.

[12] Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. Short Text Classification: A Survey. *Journal of Multimedia*, 9, 2014.

[13] Ana Cardoso-Cachopo and Arlindo L Oliveira. An Empirical Comparison of Text Categorization Methods. In *Proceedings of the International Symposium on String Processing and Information Retrieval*, 2003.

[14] Andreas Hotho, Andreas Nrnberger, and Gerhard Paa. A Brief Survey of Text Mining. *LDV Forum - Gesellschaft fur Linguistische Datenverarbeitung Journal for Computational Linguistics and Language Technology*, 20, 2005.

[15] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Machine learning: The European Conference on Machine Learning-98*, 1998.

[16] Gustavo Batista and Diego Furtado Silva. How k-Nearest Neighbor Parameters Affect its Performance. *Argentine Symposium on Artificial Intelligence*, 2009.

[17] Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.

[18] Vaishali Ganganwar. An Overview of Classification Algorithms for Imbalanced Datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2, 2012.

[19] Andrew McCallum, Kamal Nigam, et al. A Comparison of Event Models for Naive Bayes text classification. In *Proceedings of the Association for the Advancement of Artificial Intelligence: Workshop on Learning for Text Categorization*, volume 752, 1998.

[20] Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *Proceedings of the European Conference on Principle and Practice of Knowledge Discovery in Databases*, 2006.

[21] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In *Proceedings of the International Conference on Machine Learning*, 2003.

[22] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[23] M. Konchady. *Text Mining Application Programming*. Charles River Media, 2006.

[24] N. Homem and J. P. Carvalho. Authorship Identification and Author "Fingerprints". In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society*, 2011.

[25] Joao P. Carvalho, Hugo Rosa, and Fernando Batista. Detecting relevant tweets in very large tweet collections: the London Riots case study. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Fuzzy Systems*, 2017.

[26] Charu C. Aggarwal and Cheng Xiang Zhai. A Survey of Text Classification Algorithms. *Mining Text Data*, 2012.

[27] George Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3, 2003.

[28] David Martin Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation. Technical Report SIE-07-001, School of Informatics and Engineering Flinders University of South Australia, 2011.

[29] L Sorensen. User Managed Trust in Social Networking - Comparing Facebook, MySpace and Linkedin. In *Proceedings of the International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology*, 2009.

[30] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, and Francisco Carrero García. Content Based SMS Spam Filtering. In *Proceedings of the Association for Computing Machinery Symposium on Document Engineering*, 2006.

[31] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. "O'Reilly Media, Inc.", 2009.

[32] Ashraf Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial Naive Bayes for Text Categorization Revisited. In *Proceedings of the Australian Joint Conference on Advances in Artificial Intelligence*, 2004.

[33] John C. Platt. Advances in kernel methods. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. MIT Press, 1999.

[34] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 2001.

[35] Trevor Hastie and Robert Tibshirani. Classification by Pairwise Coupling. In *Proceedings of the Conference on Advances in Neural Information Processing Systems 10*, 1998.

[36] Wes McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the Python in Science Conference*, 2010.

[37] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[38] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13, 2011.