Collection and classification of telecom-related fast-text news and events

Artur Francisco Filipe Simões
artur.f.f.simoes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

July 2023

Abstract

The quality of Telecom companies' mobile networks can be seriously compromised by the occurrence of different types of events, whether they are expected or not. OSS team helps to manage the resilience and performance of the network, including Service Provisioning, Problem Management, and Quality of Service processes. Collecting probes, diagnoses, and alarms might be useful to detect the causes of the network problem. But what might have been the external cause? The cause of the cause? The purpose of the project is to automatically identify and classify events contained in short text news and civil protection occurrences from their official websites, as soon as they are known, in order to correlate them with network issues reported from the OSS team alarms. From this perspective, this thesis created an architecture that periodically collects information from the aforementioned events, in order to process it and, then, classify them using the following frameworks *Docker*, *Maven*, *Quarkus* and *Kafka*. Finally, data is stored using *PostgreSQL*. We also perform an exhaustive comparison between different lightweight models for text classification using information collected from several Portuguese online newspapers: SVM, FFP, and KNN. More complex deep models, such as BERT or RoBERTa, are dismissed due to the requirement of large amounts of training data. The proposed models predict the categories based entirely on the title and the short news snippets that are freely available. The initial findings show F1 scores exceeding 0.68 for every class, despite the dataset being unbalanced with around 14,000 samples and 4 classes.

Keywords: Short Texts classification, Support Vector Machines, Fuzzy Fingerprints, K-nearest neighbors

1. Introduction

Telecommunications service providers use Operations Support Systems (OSS) teams to develop computer systems to manage their networks. They support management and alarm functions such as network inventory, service provisioning, network configuration, and fault management. Therefore, OSS receives large amounts of alarms when monitoring events and detecting anomalous situations. Besides those alarms, there are also external factors to this monitoring system that could also be important to prevent or mitigate issues when providing telecommunications services. Namely, customer complaints about the network on social media, any type of event that jeopardizes the network service in a specific zone (such as a huge public gathering), and any type of occurrences that might compromise the good functioning of the equipment (like fires or storms). By signaling and filtering these relevant external events, the motorization of the network would be richer and would contribute to a faster fault resolution.

Specific sources of information, available in Portugal, were signaled to the context of this project: online newspapers, and Portuguese civil protection data. Social media complaints were dismissed because customer dissatisfaction is not expressed on social media scale enough to produce relevant information.

In this work, online news is used as a tool to detect possible telecom network problems by analyzing their location and classifying their topic using NLP and ML. One of the goals of this project is, based on the topic of some relevant news categories, to identify events that could compromise the quality of service of a network provider. Each piece of news should be labeled with one topic from a set of possible different ones.

Portuguese civil protection site is used to detect also some relevant events. Autoridade Nacional de Protecção Civil (ANCP), the Portuguese official civil protection authority provides public reports from all its operational data on the official website since 2007. It is possible to analyze all the occur-

rences close to real-time on Portugal's mainland. For the sake of this work, only the natures that compromise the proper functioning of the antennas are filtered. Like urban and rural fires, adverse weather conditions, the fall of buildings, trees, etc.

Part of this project consists in building a software architecture that collects, in a short period of time, data from the aforementioned sources. Then the data is processed and classified. Finally, the architecture should store the processed and classified data in a database. The architecture will be integrated into Altice's Assurance module to signal events that could jeopardize the quality of the network and correlate them with the company's official metrics.

2. Background

For years, various methods have been utilized to analyze text content and information. Text classification, an essential task in Natural Language Processing (NLP), involves categorizing text data into predefined classes or categories. This task finds applications in areas such as information retrieval, sentiment analysis, and spam filtering [7]. Text classification encompasses the construction of models that automatically classify text based on features like word frequencies, semantic meaning, and contextual information, enabling the automation of tasks that involve sifting through large volumes of unstructured text data, thereby saving time and improving efficiency.

Topic classification, a common and significant use case of text classification, entails assigning a topic or category to a given text. It finds application in domains like news categorization, document classification, and topic modeling. In news categorization, for instance, news articles are organized based on topics such as politics, business, sports, and entertainment [7].

Topic classification problems involve a concise and generic set of categories, with documents typically belonging to at least one of those categories. As the number of categories or classes increases, the difficulty of the classification task also rises exponentially. In multi-class text classification tasks, a larger number of classes necessitates more extensive training data sets. Additionally, certain classes may prove more challenging to classify due to factors like a limited number of positive training examples or a lack of effective predictive features [7].

Several widely-known methods have been commonly applied to text classification tasks. These include NB variants, KNN [4, 14], Random Forests (used for lexical resource development in sentiment analysis and opinion mining tasks) [1], Neural Networks such as Long short-term memory (LSTM) [9, 16], and SVM [8, 2, 17]. Additionally, a relatively new classifier called FFP, previously used for authorship identification [6] and Twitter topic identification [14], was employed in this project.

Recent advancements in text classification predominantly revolve around very large language models like BERT (Bidirectional Encoder Representations from Transformers) and its variants, such as RoBERTa. BERT, developed by Google, is a deep learning algorithm for pre-training in NLP. It learns general-purpose "language understanding" features through training on a large corpus of unlabeled text data, including tasks like question answering and sentiment analysis. RoBERTa, a variant of BERT developed by Facebook AI, incorporates modifications designed to further enhance performance. However, due to limitations in data availability and language mismatch, these models may not be the best fit for the current project [7].

In the realm of text classification, when lightweight models are essential, KNN and SVM are widely used and yield strong performance. Yang and Liu conducted a controlled study and reported that SVM and KNN are at least comparable to other well-known classification methods, outperforming them significantly when the number of positive training instances per topic is small [18]. SVM effectively handles unbalanced classes by maximizing the margin between them, reducing the impact of the majority class on the decision boundary. This capability proves particularly useful when instances for one or more classes are limited. KNN, being a simple and intuitive algorithm, performs well even in the presence of unbalanced classes and can effectively identify rare classes in small datasets

In this project we also use an adaptation of the FFP introduced in [6] [11], since they proved to be simple and fast. Previous works on FFP proved that this classifier outperforms SVM and KNN when there are a large number of categories. In [11], an attempt is made to classify Twitter Trending Topics into 18 broad categories, such as: sports, politics, technology, etc, and their experiments on a database of randomly selected 768 trending topics (over 18 classes) show that, using text-based and using FFP classification modeling, a classification accuracy up to 65% and 70% can be achieved, respectively.

2.1. Text Classification

Regarding text classification, the text contained in each each document, independent of its size, is the most relevant source of information. However, text is an unstructured form of data that classifiers and learning algorithms cannot directly process [5]. For that reason, our piece news must be converted into a more manageable form. A general

text analytics framework consists of three consecutive phases: Text Preprocessing, Text Representation and Knowledge Discovery, shown in Fig.1. We use three news titles as an example of Document corpus to illustrate these methods in each step:

- *"Estradas na Serra da Estrela cortadas"*
- *"Fogo volta na serra Estrela"*
- *"Aldeias da serra sem telecomunicações"*

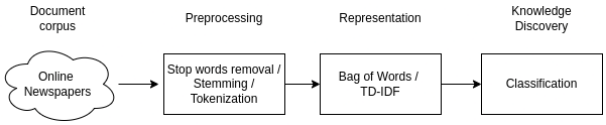

Figure 1: General Framework for Text Classification

Text preprocessing aims to make the input documents more consistent to facilitate text representation, which is necessary for most text classification tasks. Traditional text preprocessing methods include tokenization, stop word removal, stemming and lemmatization.

The output of text preprocessing for the three news are:

- *"estrad serr estrel cort"*
- *"fog serr estrel"*
- *"alde serr telecomunic"*

Text representation in NLP refers to the process of transforming text data into a numerical format that can be easily processed by machine learning algorithms. It is an important step in NLP because most machine learning algorithms require numerical data to perform classification, clustering, and other natural language processing tasks. Text representation helps to convert raw text data into a format that can be easily analyzed and processed by these algorithms. Some of the most well-known and commonly applied methods for text classification tasks include one-hot encoding, bag-of-words(BOW), word embedding, and transformer-based models.

BOW (Bag-of-Words) is a simple and widely used technique for text representation [17, 14, 15]. It involves the following steps: (1) creating a vocabulary of unique words from the text, (2) assigning a unique index to each word in the vocabulary, and (3) creating a vector for each document where each key corresponds to a word in the vocabulary, and the value represents the count of the word in the document.

The advantages of BOW include its simplicity and ability to capture word frequency and distribution, making it useful for text classification. However, BOW has limitations: it disregards word order and context, treats all words equally, and can be sensitive to noise or irrelevant words, affecting representation accuracy.

To enhance BOW, it can be combined with TF-IDF (Term Frequency-Inverse Document Frequency) [17, 14, 15]. TF-IDF adjusts the weights of words based on their frequency in the document and their occurrence in the corpus. This normalization technique assigns higher importance to relevant words in the document and reduces the impact of common words across the corpus.

In practice, after applying the three steps of BOW, each document is represented by a feature vector. To improve the representation, normalization techniques like TF-IDF can be applied. TF-IDF converts the count features into floating-point values suitable for classification tasks.

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

$$idf = \log \frac{N}{df(w)} \tag{2}$$

where:

– tf(w) is term frequency (the number of word occurrences in a document) – df(w) is document frequency (the number of documents containing the word)

– N is number of documents in the corpus

– tfidf(w) is the relative weight of the feature in the vector

As succinctly explained in [12], TF-IDF assigns a weight to a term in document that is: 1) highest when the term occurs many times within a small number of documents; 2) lower when the term occurs fewer times in a document, or occurs in many documents; 3) lowest when the term occurs in virtually all documents, which is the case of stop words, that tend to 0 [13];

Using BOW to model the three messages with a TF-IDF weight, the corpus can be represented as a words * documents matrix. Each row represents a word (7 distinct words in total) and each column represents a message, as shown below:

$$\begin{bmatrix} \textit{estrad} \\ \textit{serr} \\ \textit{estrel} \\ \textit{cort} \\ \textit{fog} \\ \textit{alde} \\ \textit{telecomunic} \end{bmatrix} = \begin{bmatrix} 0.477 & 0 & 0 \\ 0 & 0 & 0 \\ 0.176 & 0.176 & 0 \\ 0.477 & 0 & 0 \\ 0 & 0.477 & 0 \\ 0 & 0 & 0.477 \\ 0 & 0 & 0.477 \end{bmatrix} \tag{3}$$

Finally, after the feature vector is created, it can be used to represent the text as a BOW. The feature vector can be passed as input to a machine learning model or used to compute the similarity between different pieces of text applying methods like classification or clustering.

2.2. Support Vector Machine

When using SVM the similarity measure is called kernel trick to split data, KNN uses Euclidean distance to get the *k* nearest neighbours and FFP computes a similarity score.

SVM are commonly used for Topic Classification tasks [2] [17] [3]. This classification is based on set of hyperplanes in a high-dimensional space. In a *p* dimensions space the hyperplanes are characterized by $p-1$ dimensions. equation 4 describes the hyperplanes defined in Figure 2.

$$H(\vec{x}) = \vec{w}^T * \vec{x} + w_0 \tag{4}$$

The Figure 2, represents three different hyperplanes and functional margins: the hyperplane H_1 does not separate the positive from the negative instances.H_2 does, but it does not guarantee the maximum distance between them. Finally, H_3 offers the necessary solution as *z3* represents the largest functional margin.

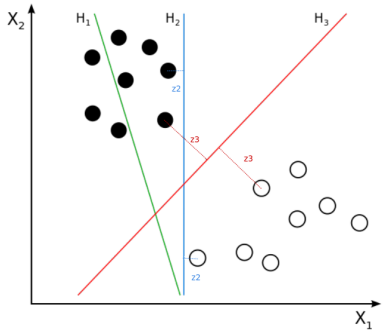

Figure 2: 2D Support Vector Machines classifier with three hyperplanes

A good separation is achieved by the hyperplane that has the largest functional margin, the distance to the nearest training data point of any class (represented by *z*),

$$z = \frac{H(\vec{x})}{|\vec{w}|} \tag{5}$$

When building a SVM classifier, the goal is to minimize $|\vec{w}|$, in order to obtain the largest functional margin.

In general, a larger margin means a lower classifier generalization error. SVM can efficiently perform linear and non-linear classifications using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

If it is impossible to find an optimal decision boundary using the aforementioned kernel trick, the SVM algorithm will still try to find a decision boundary that separates the data as well as possible. However, the resulting decision boundary may not be optimal and could potentially misclassify some data points.

An interesting property of SVM is that the decision surface is determined only by support vectors, which are the only effective elements in the training set; if all other points were removed, the algorithm will learn the same decision function. This characteristic makes SVM theoretically unique and different from other methods where all the data points in the training set are used to optimize the decision function [18].

2.3. K-Nearest Neighbours

KNN is an example-based classifier, commonly referred in the literature [7] [3] [10]. The representations of training data (usually TF-IDF representations) are simply stored together with their category labels. In the figure 3, represents a decision whether a new document represented by the yellow dot belongs to a category blue, KNN checks if the k training documents closest to the new document belong to blue. If the answer is positive for a sufficiently large proportion of documents then the result is positive.

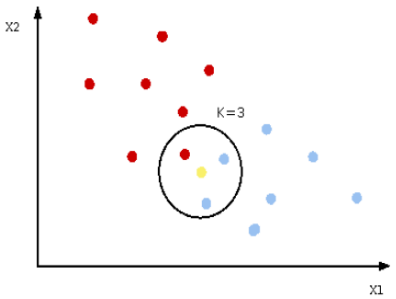

Figure 3: 2D k-Nearest Neighbours classifier

Usually, when testing the value of k, we could start with the square root of the total number of data points. It is also a good practice to choose an odd value of k to avoid ties.

In order to find the k new document nearest neighbours, the classifier computes the distance between the new document and each one of the training documents using Minkowski Distance, given by the following equation

$$distance = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{6}$$

Minkowski distance is the generalized distance metric. Here generalized means that we can manipulate the above formula to calculate the dis-

tance between two data points in different ways. In the aforementioned example, p=2, then the distance correspond to Euclidean Distance.
In the context of text classification using KNN with Minkowski distance, each document is represented as a vector of features, where each feature represents a term or word. The dimensionality of the vector is equal to the total number of unique terms across all documents in the corpus. The resulting distance value represents the similarity between the two documents, with smaller distances indicating greater similarity. In text classification using KNN with Minkowski distance, the K nearest neighbors are selected based on their Minkowski distance from the test document, and the majority class among the K neighbors is assigned to the test document as its predicted class.

KNN is known to be affected by noisy data, still it is considered one of the simplest and best performing text classifiers, whose main drawback is the relatively high computational cost of classification, because for each test document it must compute its similarity to all of the training documents. The training is fast, but the classification is slow because it implies computing all the similarities between a document that has not been categorized and the existing a collection of training documents. However, given that in our work we are using short text snippets, KNN should be adequate at least as a baseline while the dataset is in its earliest stages.

2.4. Fuzzy Fingerprint

The Fuzzy Fingerprint Classifier is a machine learning model used for text classification tasks such as authorship identification [6] and topic identification in Twitter [14]. The classifier utilizes two hyperparameters: *K* for the fingerprint size and *threshold* for the minimum classification score.

The workflow of the Fuzzy Fingerprint Classifier can be summarized as follows:

- Initialize the classifier with a list of topics and create an empty fingerprint dictionary for each topic.
- Process the input text, adding its words to the corresponding topic's fingerprint.
- Compute the TF-IDF for the top-k word frequencies in all known texts of each news topic.
- Build the fuzzy fingerprint for each topic using a Pareto distribution on the top-k word frequencies.
- Predict the news topic by comparing the input text's fingerprint with each topic's fingerprint and selecting the most similar one.

To build the fingerprint library, the proposed method considers the most frequent words in the news and accounts for their TF-IDF values. The fingerprint consists of a *k*-sized bi-dimensional array containing the top-*k* words and their membership values based on the Pareto distribution. The membership equation is defined as follows:

$$\mu_{ab}(i) = \begin{cases} 1 - (1-b)\frac{i}{kb}, i < a \\ \frac{a(1-\frac{i-a}{k-a})}{k}, i \geq a \end{cases} \quad (7)$$

In Equation 7, *a* and *b* are constants that control the slope of the tail of the distribution. Larger values of *a* and *b* result in a steeper tail.

The similarity score used is the Tweet-Topic Similarity Score (T2S2) [14], which calculates the relevance of a news snippet to a topic. The T2S2 equation is as follows:

$$T2S2(\Phi, T) = \frac{\sum_v \mu_\Phi(v) : v \in (\Phi \cap T)}{\sum_{i=0}^{j} \mu_\Phi(w_i)} \quad (8)$$

In Equation 8, Φ represents the topic fingerprint, *T* is the set of words in the news snippet, $\mu_\Phi(v)$ is the membership degree of word *v* in the topic fingerprint, and *j* is the number of features of the news. The score ranges between 0 and 1, with values closer to 1 indicating a higher similarity.

By setting a threshold value between 0 and 1, news snippets with a similarity score below the threshold are considered irrelevant.

In summary, the Fuzzy Fingerprint Classifier applies a Pareto-based fingerprinting approach and utilizes the T2S2 similarity score for text classification tasks. The classifier has shown effectiveness in authorship identification [6] and topic identification in Twitter [14].

3. Methodology

The architecture in Figure 4 was designed to fulfill the following processes: data collection, data processing, and data insertion in the database. It consists of two pipelines, each per data source type, online newspapers news, and civil protection occurrences. These data sources already exist, so no web scraping is needed. The goal of this architecture is to enrich a database with news or civil protection occurrences. Those should be well-defined, located, and classified so can be viewed and compared with network metrics and alarms resulting from other companies' products. Our model aims to be fast, robust, and fault-tolerant (due to Kafka implementation). Since the online news pipeline is the most complex one and is more relevant to the analysis, it is explained in detail in the following subsection. However, the Civil protection data pipeline follows a similar process.

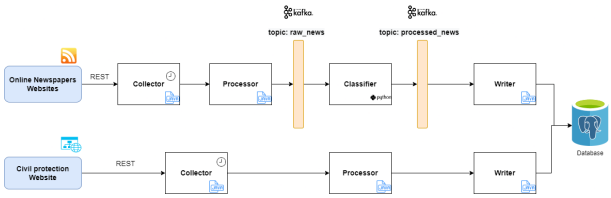

Figure 4: Proposed architecture

3.1. System Architecture

The ETL (Extract, Transform, Load) pipeline for online news consists of several modules: Collector, Processor, Classifier, Writer, and Database.

The Collector module retrieves data from seven different Portuguese online newspapers' APIs, using RSS technology for most sources. The data is collected in real-time through REST GET calls every 25 minutes, fetching approximately 15 news articles per source in XML format. The Collector module ensures a maximum collection time of 17 minutes and 30 seconds per hour.

The Processor module, implemented in Java, consumes data from the Collector module and filters and standardizes the raw data. Irrelevant attributes are removed, and the remaining attributes include Title, Description, Link, and Publication Date. The processed news snippets are then sent to the Classifier module via a Kafka channel in JSON format.

The Classifier module analyzes the text of the news snippets received from the Processor module. It consumes Kafka streams from the "raw news" topic, deserializes the JSON messages, and concatenates the news title and description for analysis. The module uses SVM multiclass classification to predict the news category and employs Named Entity Recognition (NER) to extract location information from the text.

The Writer module consumes JSON streams from the "processed news" topic, produced by the Classifier module. It creates instances of a Java class to store the objects from the Kafka stream. This module is crucial as it ensures data persistence and prevents data loss, even if the database insertion fails. Kafka temporarily stores the streams before their insertion into the database.

The data is ultimately stored in a PostgreSQL database, serving as the persistent storage for the processed news articles.

The described ETL pipeline for online news was deployed in AlticeLabs' virtual machines, and the project was automated to achieve this goal. The implementation in virtual machines involved deploying the various modules in the cloud, creating Kubernetes containers managed with Docker, and generating Quarkus images.

3.2. Dataset Information

For classifier training purposes, each news snippet record contains text and a topic. All the news used to train and test the different classifiers was retrieved daily from several Portuguese online newspapers from December 29^{th}, 2021 until July 25^{th}, 2022, containing a total of 14185 up-to-date news. Commonly, each text consists of only a couple of sentences and has on average 37.2 words, including stopwords, and has on average 22.7 words, excluding stopwords. The reduced number of words makes the text based classification task a difficult problem. Figure 5 shows a histogram of the frequency of news per topic. Revealing that the data set is highly unbalanced. There are 14185 records with the following distribution *fires*: 818, *meteorologic*: 429, *public gatherings*: 369 and *other*: 12570. The minor category *public gatherings* is assigned to only 2.6% of the records and the largest category *others* is assigned to 88.6% of the records. It is important to note that it was decided to keep the real-world dataset unbalanced in order not to bias the models. This allows us to expect a similar model performance when under real use cases.

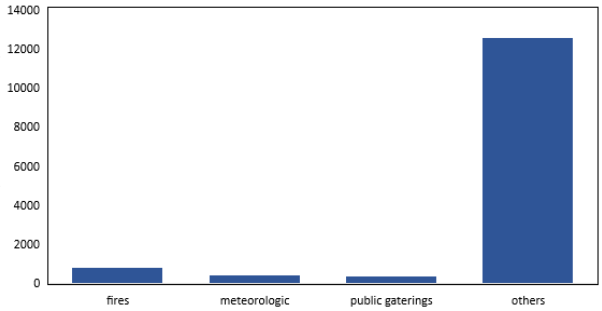

Figure 5: Number of news labeled for a given category.

Text classification works usually demand higher amounts of data to achieve better and more reliable results. Moreover, data was collected over half a year and it is biased by the conditions of that period. During the first times of collection, the Covid pandemic was in every piece of news, later the Ukrainian war and the latest the huge wildfires in Portugal. Certainly, no two years are the same, but it is expected that the data collected does not represent the generality of the Portuguese national news. Nevertheless, the chosen classification model is expected to improve its performance over time as the data set increases.

3.3. Evaluation Metrics

The metrics used to evaluate our classification results were: Precision, Recall, and F1-Measure, computed using *sklearn.metrics* Python library.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1_{score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

In the context of classification metrics, micro-averaged and macro-averaged scores are two ways to calculate a model's overall performance.

Micro-averaged scores aggregate individual contributions from each instance in the dataset. This approach assigns equal weight to each instance, tends to be influenced by the classifier's performance on common categories. On the other hand, macro-averaged scores compute the metric separately for each class and then average the values across all classes. This method assigns equal weight to each class, and places more emphasis on small categories.

For classification purposes, the dataset was divided into a training set (80% of the records) and a test set (20% of the records). The training set was utilized to tune hyperparameters and test various text-processing techniques, while the test set was kept aside for blind evaluation of the classifiers.

The evaluation process involved the following steps: optimizing hyperparameters on the validation set, applying the best combination of hyperparameters and text-processing techniques to the test set, and recording the results in a confusion matrix. The confusion matrix was then analyzed to assess the overall performance of the classifiers and identify areas for potential improvement.

SVM and KNN models were implemented using the scikit-learn Python library. The FFP method followed the approach outlined in [14]. All methods utilized the same training and test datasets.

To achieve satisfactory results, a larger dataset was required. To overcome this limitation, 5-fold cross-validation was employed. The data was split into training sets (80%) and test sets (20%) to align with the 5 cross-validation proportions. This procedure facilitated hyperparameter tuning and mitigated overfitting concerns.

4. Results & discussion

The evaluation process involved the following steps: First, the hyperparameters were optimized on the validation set. Then, the best combination of hyperparameters and text processing techniques was applied to the test set (blind test), and the results were recorded in a confusion matrix. Finally, the confusion matrix was analyzed to assess the overall performance of the classifiers and to identify any potential areas for improvement.

4.1. Support Vector Machine

Both multiclass and binary class SVM models were trained using a bag-of-words representation with TF-IDF weighting, and the soft-margin parameter C that was optimized during the training phase. The SVM models were tested with both linear and non-linear kernels, and it was found that the linear kernel provided better results. Word embeddings were tested but produced worse results, probably due to the lack of data to create a good model.

For the multi-class model, we tried to predict all classes from the same SVM model. During the training phase, hyper-parameters C and kernel were tuned to get the best results. The algorithm was trained with a C value in the range of 0.001 to 1000, the kernels tested were linear, poly, rbf, and sigmoid, and different text processing techniques were tested. Table 1 resumes the algorithm's performance with the influence of stemming and removing stopwords when tuning hyperparameters on the training set. The hyperparameters that produced the best results are linear *kernel*, *C* equals to 1, and stemming and removing stopwords as text processing techniques.

Table 1: Hyper parameters tuning and various text processing techniques influence the training set on multi-class SVM.

Kernel	C	Text Processing	Precision	Recall	F1-Score
linear	2	None	0.818	0.668	0.735
linear	2	Remove stopwords	0.834	0.642	0.726
linear	1	Stemming	0.818	0.666	0.734
linear	1	Stemming, Remove stopwords	0.795	0.685	**0.736**

In Table 2, the results of SVM multi-class classification are presented, using the hyperparameters and text processing techniques that had the best F1 scores during the training phase. Figure 6 resumes the confusion matrix of the SVM multi-class performance.

Table 2: SVM multi-class scores on the test set for linear Kernel, C = 1, text with no stopwords and stemming for each topic

Topic	Precision	Recall	F1-Score	Support
Fires	0.87	0.86	0.865	164
Meteorologic	0.80	0.73	0.763	86
Public gatherings	0.80	0.49	0.608	74
Macro avg	0.823	0.693	0.745	324

Predicted topic

True topic	Fires	Meteorologic	Public Gatherings	Others
Fires	141	1	0	22
Meteorologic	3	63	0	20
Public Gatherings	0	0	36	38
Others	18	15	9	2472

Figure 6: Confusion matrix results for multi-class SVM with Stemming, Remove stopwords and C=1

The SVM binary model was constructed from four binary models corresponding to one of our

classes. To train each model we selected news labeled with the corresponding category as positive samples and all the other news as negative samples. At the end of the classification, each piece of news is assigned to the category with the highest score.

For the binary-class model, we used the same tuning as multi-class SVM. The algorithm was trained with a C value in the range of 0.001 to 1000, all kernels were tested, and also different text processing techniques. Table 3 resumes the binary SVM performance. The results obtained for SVM binary classifier were similar to the SVM multi-class. The hyperparameters that produced the best results are linear *kernel*, *C* equals to 1, and removing stopwords as text processing techniques.

Table 3: Hyper parameters tuning and various text processing techniques influence the training set on binary-class SVM.

Kernel	C	Text Processing	Precision	Recall	F1-Score
linear	0.001	None	0.833	0.629	0.717
linear	1	Remove stopwords	0.799	0.681	**0.735**
linear	1	Stemming	0.774	0.657	0.711
linear	1	Stemming, Remove stopwords	0.778	0.693	0.733

In Table 4, the results of SVM binary-class classification are presented, using the same approach as SVM multi-class. Figure 7 resumes the confusion matrix of the SVM binary performance.

Table 4: SVM binary-class scores on the test set for linear Kernel, C = 1, text with no stopwords for each topic

Category	Precision	Recall	F1-Score	Support
Fires	0.870	0.86	0.865	164
Meteorologic	0.797	0.690	0.745	86
Public gatherings	0.760	0.530	0.620	74
Macro avg	0.810	0.693	0.743	324

Predicted topic

True topic	Fires	Meteorologic	Public Gatherings	Others
Fires	141	2	0	21
Meteorologic	3	59	0	24
Public Gatherings	0	0	39	35
Others	18	13	12	2471

Figure 7: Confusion matrix results for binary SVM with Remove stopwords and C=1

4.2. Fuzzy Fingerprints

In our study, we employed two distinct methodologies for classifying the FFP. The first methodology involved training the model using four news topics, whereby each news instance was assigned a similarity score by each fingerprint corresponding to each topic. If the similarity score was below the predefined threshold, the instance was classified as *others*. In contrast, the second methodology entailed testing the model with only three topics, excluding the *others* category. Any news instance with a similarity score below the threshold was attributed to the *others* topic. After conducting tests, it was revealed that the first approach yielded a 3% superior F1-score, thus rendering it the approach that is expounded upon in the ensuing text. For the FFP model, we tried to predict all classes from the same model. During the training phase, hyperparameters *K* and *threshold* were tuned to get the best results. The algorithm was trained with a *K* value in the range of 150 to 800, the *threshold* in the range of 0.08 to 0.24 and different text processing techniques were tested. Table 5 resumes the algorithm's performance with the influence of stemming and removing stopwords when tuning hyperparameters on the training set. The hyperparameters that produced the best results are *K* equal to 500, *threshold* equal to 0.13, and removing stopwords as text processing techniques.

Table 5: Hyper parameters tuning and various text processing techniques influence the training set on FFP.

K	Threshold	Text Processing	Precision	Recall	F1-Score
600	0.09	None	0.510	0.831	0.632
500	0.13	Remove stopwords	0.606	0.743	**0.668**
500	0.12	Stemming	0.51	0.789	0.620
300	0.14	Stemming, Remove stopwords	0.567	0.732	0.639

In Table 6, the results of FFP model classification are presented, using the hyperparameters and text processing techniques that had the best F1 scores during the previous phase. Figure 8 resumes the confusion matrix of the FFP performance.

Table 6: FFP scores on the test set for threshold=0.13, K=500, text with no stopwords for each topic

Category	Precision	Recall	F1-Score	Support
Fires	0.740	0.830	0.782	164
Meteorologic	0.650	0.850	0.737	86
Public gatherings	0.460	0.680	0.550	74
Macro avg	0.617	0.787	0.690	324

Predicted topic

True topic	Fires	Meteorologic	Public Gatherings	Others
Fires	136	2	0	26
Meteorologic	2	73	0	11
Public Gatherings	1	0	50	23
Others	45	37	59	2373

Figure 8: Confusion matrix results for FFP threshold=0.13, K=500, text with no stopwords

4.3. K-Nearest Neighbours

During the training phase, hyper-parameter K was tuned to get the best results. The algorithm was trained with a K value in the range of 1 to 201 and different text processing techniques were tested. Table 7 resumes the algorithm's performance with the influence of stemming and removing stopwords when tuning hyperparameters on the training set.

The hyper-parameter that produced the best results is K equals 11 and removing stopwords as text processing techniques.

Table 7: Hyper parameters tuning and various text processing techniques influence the training set on KNN.

K	Text Processing	Precision	Recall	F1-Score
17	None	0.783	0.622	0.693
11	Remove stopwords	0.774	0.657	**0.711**
51	Stemming	0.867	0.5	0.634
41	Stemming, Remove stopwords	0.865	0.551	0.673

Overall results with K equal to 11, with stopwords removed, are shown in Table 8. Figure 9 resumes the KNN performance.

Table 8: KNN scores on the test set for K=11, text with no stopwords for each topic

Category	Precision	Recall	F1-Score	Support
Fires	0.830	0.830	0.830	164
Meteorologic	0.790	0.730	0.759	86
Public gatherings	0.760	0.350	0.480	74
Macro avg	0.793	0.637	0.689	324

		Predicted topic			
		Fires	Meteorologic	Public Gatherings	Others
True topic	Fires	136	2	0	26
	Meteorologic	6	63	0	17
	Public Gatherings	0	0	26	48
	Others	22	15	8	2469

Figure 9: Confusion matrix results for KNN, K=11, text with no stopwords

4.4. Classifiers comparison

We have performed experiments using four classification approaches, multi-class and binary SVM, KNN, and FFP. Our real-world dataset is highly unbalanced. Our results reveal that despite the short text size (title and description of a piece of news) it is possible to predict its category with an overall performance of about 74.5% F1-Score when using a binary SVM approach on Figure 9.

The public gatherings topic, in particular, exhibits lower performance across all models. This can be attributed to several factors:

- Fewer samples: The public gatherings topic has the fewest samples, which might not be enough to train a robust model.
- Heterogeneity: This class encompasses a diverse range of events and scenarios. This can make it difficult for the model to accurately identify and classify these events.
- Complexity: Public gatherings can also have complex events that involve a lot of different factors, such as the number of people, location, and purpose.

Overall, the poor performance of the public gatherings class is likely due to a combination of these factors. Including confusion matrices in the analysis enables realize how frequently this topic in misclassified as the topic ”Others” due to its complexity and different purposes. To improve this topic performance, one could consider collecting more data, creating a more homogeneous class, or using a more complex model to better capture the nuances of public gatherings.

When comparing all methods, it is evident that the SVM models outperform the others. Regarding the referenced evaluation metrics, the SVM binary model is slightly better than the SVM multi-class. The FFP strategy, while achieving the best recall from all the classifiers achieves 21% lower precision in comparison to the SVM multi-class. Classifiers KNN and FFP have the same F1 score, although both precision and recall values are contrasting. FPP performance is limited by the number of classes, because when dealing with a large number of classes is achieves good performance. One reason that can explain why SVM performs better than KNN is that it can be better fine-tuned than the latter. SVM models are often good at handling imbalanced datasets, where the number of samples in each class is not equal. This is because the SVM algorithm is not affected by the imbalance of the data and instead focuses on finding the best separation boundary between the classes. Overall, the SVM model's ability to handle imbalanced data and the flexibility in hyperparameter tuning may have contributed to its high F1 score in this classification.

Table 9: Comparison of all models using Precision, Recall, and F1-Score

Model	Precision	Recall	F1-Score
SVM multi-class	**0.823**	0.693	**0.745**
SVM binary	0.810	0.693	0.743
FFP	0.617	**0.787**	0.690
KNN	0.793	0.637	0.689

5. Conclusions

Our project adopts a heuristic approach informed by understanding of the needs of assurance applications. By providing a detailed overview of requirements and data characteristics, along with statistical analysis, each model is evaluated regarding its strengths and weaknesses across text classification. F1-score is used as an evaluation metric to gain a better assessment of topic properties and distribution. The study reveals that the quality and quantity of training data can significantly impact classification performance.

In addition, this thesis demonstrates that despite the challenges posed by short text length, it is possible to achieve an overall performance of approximately 74.5% F1-score for predicting news categories using a binary SVM approach. Overall, this

research provides valuable insights into the challenges and opportunities of text classification for news data in Portuguese.

This study employed a specific methodology to collect data, which allowed for the reflection of potential issues faced by an operator in a particular region. However, it is important to acknowledge the limitations and areas for improvement in the current approach.

To address this limitation and improve the classifier's performance, future work could explore the collection of additional data. Increasing the amount of data available for analysis could enhance the model's ability to identify and classify network-related issues. Furthermore, efforts should be made to ensure that the collected data is more homogeneous in terms of classes. By creating more balanced and representative classes, the classifier can better generalize and accurately classify network problems.

In conclusion, while this study successfully applied the methodology to identify and prioritize network issues, there are limitations that need to be addressed in future work. By collecting more data and creating more homogeneous classes, the classifier's performance can be improved, leading to more accurate identification and classification of network problems. Furthermore, exploring the impact of identified events on service complaints will provide valuable insights for enhancing customer satisfaction and meeting market demands.

References

[1] E. A. . S. F. Baccianella, S. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, pages 2200–2204, 2010.

[2] R. R. Batista, F. Sentiment analysis and topic classification based on binary maximum entropy classifiers. *Procesamiento de Lenguaje Natural*, 50:77–84, 03 2013.

[3] O. A. L. Cardoso-Cachopo, A. An empirical comparison of text categorization methods. In M. A. Nascimento, E. S. de Moura, and A. L. Oliveira, editors, *String Processing and Information Retrieval*, pages 183–196, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] K. D. A. M. DAha, D. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[5] S. J. Feldman, R. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, volume 34. 12 2006.

[6] J. Homem N., Carvalho. Authorship identification and author fuzzy "fingerprints". In *Fuzzy Information Processing Society (NAFIPS), 2011 Annual Meeting of the North American*, 03 2011.

[7] K. S. T. V. Ikonomakis, E. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4:966–974, 08 2005.

[8] e. a. Lee, K. Twitter trending topic classification. pages 251–258, 12 2011.

[9] C. G. W. X. . Y. D. Li, X. Acoustic modeling using deep neural networks for lvcsr. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4445–4449, 2015.

[10] H. S. Lim. Improving knn based text classification with well estimated parameters. In N. R. e. a. Pal, editor, *Neural Information Processing*, pages 516–523, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[11] C. J. e. a. Marujo, L. *Textual Event Detection Using Fuzzy Fingerprints. In: Angelov, P., et al.* 2015.

[12] U. J. Rajaraman, A. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[13] S. P. . J. W. Riloff, E. Feature subsumption for opinion analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2006.

[14] B. F. C. J. Rosa, H. Twitter topic fuzzy fingerprints. pages 776–783, 07 2014.

[15] e. a. Sood, S. Tagassist: Automatic tag suggestion for blog posts. In *ICWSM*, 2007.

[16] . L. Q. Vinyals, O. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[17] M. C. Wang, S. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[18] L. X. Yang, Y. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 42–49, New York, NY, USA, 1999. Association for Computing Machinery.