Relevance classification on service desk texts using Natural Language Processing

Marciel Mario Degasperi[1], Daniel Cavalieri[1], and Fidelis Zanetti de Castro[1]

[1]Federal Institute of Education Science and Technology of Espírito Santo at Serra Espirito Santo

July 6, 2023

Abstract

Service desk systems have a vast and rich base of information, consisting of the history of calls made, which can and should be used as a reference base for subsequent calls. Common search tools, such as keyword searches, prove to be unfeasible in large datasets, in addition to being able to bring results not necessarily related to the problem. "State-of-the-art" techniques exist, but they require high computational and operational costs for their training and use. In this sense, the purpose of this work is to investigate the sensitivity of machine learning algorithms in finding the characteristic defined here as "relevance": the characteristic of texts with knowledge that can be reused. The motivation is that non-relevant texts can be removed in advance from the database, allowing complex algorithms to be employed in a more condensed database, reducing computational costs. Tests were performed with several combinations between the TF-IDF vectorizer and the word embedding Doc2Vec and the classic classifiers Naive-Bayes, Adaptive Boosting, Random Forest, Stochastic Gradient Descent, Logistic Regression, Support Vector Machine and Light Gradient Boosting Machine, and the classifier TextConvoNet, an architecture based on Convolutional Neural Networks. The TextConvoNet classifier presented the best results, with metrics close to 0.93, showing that the concept is detectable and that the technique is viable for removing non-relevant texts from a database.

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ORIGINAL ARTICLE

Relevance classification on service desk texts using Natural Language Processing

Marciel Mario Degasperi | Daniel Cruz Cavalieri | Fidelis Zanetti de Castro

[1]Department of Control and Automation Engineering, Federal Institute of Education, Science and Technology of Espírito Santo at Serra Espirito Santo, Brazil

Correspondence
Marciel Mario Degasperi.
Email: marciel.deg@gmail.com

Summary

Service desk systems have a vast and rich base of information, consisting of the history of calls made, which can and should be used as a reference base for subsequent calls. Common search tools, such as keyword searches, prove to be unfeasible in large datasets, in addition to being able to bring results not necessarily related to the problem. "State-of-the-art" techniques exist, but they require high computational and operational costs for their training and use. In this sense, the purpose of this work is to investigate the sensitivity of machine learning algorithms in finding the characteristic defined here as "relevance": the characteristic of texts with knowledge that can be reused. The motivation is that non-relevant texts can be removed in advance from the database, allowing complex algorithms to be employed in a more condensed database, reducing computational costs. Tests were performed with several combinations between the TF-IDF vectorizer and the word embedding Doc2Vec and the classic classifiers Naive-Bayes, Adaptive Boosting, Random Forest, Stochastic Gradient Descent, Logistic Regression, Support Vector Machine and Light Gradient Boosting Machine, and the classifier TextConvoNet, an architecture based on Convolutional Neural Networks. The TextConvoNet classifier presented the best results, with metrics close to 0.93, showing that the concept is detectable and that the technique is viable for removing non-relevant texts from a database.

KEYWORDS:
Classification, Natural Language Processing, Service Desk.

1 | INTRODUCTION

The concern with customer satisfaction is a constant for companies to remain competitive. This satisfaction occurs through the evaluation that the consumer makes in relation to the product or service offered. Failure to meet the needs and expectations of the customer generates dissatisfaction with the services and, consequently, with the company[1]. In order to stand out from the rest and generate credibility, it is essential that organizations have the ability to promptly respond to requests, suggestions or complaints[2].

Service Desk is a term that designates the customer support service for support and resolution of technical problems[3]. Service Desk systems play an important role as intermediaries between companies and their customers. They offer an efficient system that allows the collection and quick resolution of problems, as well as registering the solutions presented and correlating them with other situations[4].

In a typical Service Desk system, the flow starts with a service ticket (*ticket*) created by a user, detailing the problems encountered. Each *ticket* is manually associated with a category, which is linked to one or more knowledge bases. Finally, a solution is retrieved from a knowledge base related to the problem. This cycle can be repeated until the user has his request answered or his problem solved[5].

Service desk systems act as a link between companies and customers through a system that enables the compilation and resolution of problems, as well as records of the solution presented and correlation with other situations[4]. With this, an extensive database is created for the company, managing problems, solving them at their root, and reducing operational costs. Thus, service desk systems can centralize various information for several service areas, becoming a pivotal point in administration and problem solving[6].

With the growth of the service base over time, the search for information becomes complex. A simple way to get a set of responses to a user query is to determine which documents in a collection contain a particular set of query keywords. However, this may not be enough to satisfy the user, as the presence of non-relevant documents among the documents returned by a query is practically inevitable. In this scenario, the main objective of these systems should be to retrieve as many relevant documents as possible and as few non-relevant documents as possible[7]. In this scenario, there is a need to use support tools to analyze attendances to help classify and recover information in the dataset. As these data are basically composed of descriptive texts, there is a need to propose and implement computational systems based on natural language processing.

Service Desk systems generate a rich but sparse knowledge base. The services provided may not contain knowledge that can be reused in the future. For example, a report of a one-off problem that was corrected immediately, a call regarding an overdue charge, or a call opened in error. A characteristic of this type of system is that not all the calls made have information that can be reused to assist in handling new calls. This data harms search algorithms, whether a simple keyword search algorithm or complex algorithms with neural networks.

This work aims to examine the feasibility of existing classification algorithms in the literature for the task of identifying services that possess reusable knowledge, referred to as "relevant" in this context. By obtaining these results, it becomes possible to compile a new, more condensed dataset, allowing for the optimal application of other knowledge extraction strategies.

The concept of relevance is a widely debated topic in Information Science. For[7], relevance is defined as a quality metric, linked to the satisfaction of a document meeting a given request. The notion of relevance is human and not technical, composed of a range of variables and, therefore, fragile and difficult to define.

The concept of relevance explored in this work is broader and abstract. In this study, relevance was defined as the characteristic of texts that have any reusable knowledge. In other words, a text is defined as "relevant" if a reader can learn something from reading it and reuse this knowledge in the future, regardless of the area of knowledge correlated to the text. A person reading a text on any subject can say with some ease "I learned something from reading this" or "There is nothing important in this document", although it is difficult for a person to describe what textual elements were used for this statement.

In this study, the concept of relevance takes on a broader and more abstract interpretation. Here, relevance is defined as the characteristic of texts containing any form of reusable knowledge. In other words, a text is considered "relevant" if a reader can acquire new knowledge from reading it and subsequently apply that knowledge in the future, regardless of the specific domain associated with the text. When reading a text on any subject, an individual can easily determine whether they have learned something valuable or if the document lacks important information. However, articulating the specific textual elements used to make such judgments can be difficult for the individual to express.

2 | LITERATURE REVIEW

In this section we will present a brief literature review of the concepts used in this research.

2.1 | Natural Language Processing

Natural Language Processing (NLP) is an important area of interest in Artificial Intelligence and Computer Science. The importance of this field of study grows in parallel with the unprecedented amount of textual information digitized in history. As a scientific field of study, NLP encompasses concepts from computer science, linguistics and mathematics with the main objective of translating human language into commands that can be executed by computers. While there are several well-established NLP

techniques, natural language remains difficult to process. As an example, it is possible that omitted words and bad grammar do not interfere with communication between two people. For computers, however, this is a big obstacle[8].

NLP can be divided into four stages: pre-processing, vectorization, model training and model evaluation. Text pre-processing aims to obtain "clean" text, removing symbols and other elements that may generate noise in subsequent analysis. After pre-processing the text, an algorithm is selected that converts words into a numerical representation, such as vectors. Based on these word vectors, algorithms can be applied to train a model that can solve the real problem. Finally, after training the model, it is necessary to evaluate its quality and its applicability in other scenarios[8].

2.2 | Numerical representation of textual content

Although computers possess the ability to handle various multimedia tasks, they fundamentally operate as machines that execute instructions based on binary numbers. Therefore, in order to utilize computational resources for text classification, it becomes necessary to employ strategies that can represent sets of words in a numerical format while preserving their essential characteristics[9]. Two significant techniques in machine learning, namely vectorization and word embedding, play a crucial role in enhancing the performance and efficiency of natural language processing tasks.

Vectorization involves the conversion of textual data into numerical representations that can be easily manipulated by algorithms. Word embedding, on the other hand, is a specific type of vectorization that assigns high-dimensional vectors to words, capturing both their semantic and syntactic similarities. Through the utilization of vectorization and word embedding, we can effectively reduce the dimensionality and sparsity of the data, allowing models to learn from the contextual information embedded within the words.

2.2.1 | Term Frequency times Inverse Document Frequency (TF-IDF)

A common way of categorizing texts is by identifying the occurrence of keywords that characterize documents on a specific topic. For example, baseball articles would tend to have many occurrences of words like "ball", "bat", "throw", "run", and so on. The algorithm *Term Frequency times Inverse Document Frequency* (TF-IDF) is a vectoring algorithm used to calculate the importance of a word in a document relative to a set of documents[10]. Thus, it is possible to replace each word of a text with a number that indicates its importance in the document context.

The algorithm uses two measures: the term frequency (TF) and the inverse document frequency (IDF). Term frequency (TF) is the measure of how many times a term appears in a document, defined by Equation 1:

$$TF_{ij} = \frac{f_{ij}}{max_k(f_{kj})} \tag{1}$$

where

- f_{ij}: Number of occurrences of the term i in the text j;
- $max_k(f_{kj})$: Largest index k of the words in the text j, that is, the total number of words in the text j.

The inverse document frequency (IDF) is the measure of how rare a term is in relation to a set of documents, defined by Equation 2:

$$IDF_i = \log_2 \frac{N}{n_i} \tag{2}$$

where

- N: Total number of documents;
- n_i: Number of documents containing the term i.

The importance of a term in a document is given by multiplying the TF and IDF measures[10]. Therefore, the TF-IDF value for each word i of the document j is given by Equation 3:

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i \tag{3}$$

A term's importance to a document increases if it appears frequently in the document and rarely appears in other documents. This allows the TD-IDF algorithm to be able to highlight relevant keywords in a document, which is useful in information retrieval and document indexing tasks[10].

2.2.2 | Word2Vec

Word2Vec (Word to Vector) is an word embedding algorithm for generating vector representations of words, or embeddings. It was developed in 2013 and is widely used in natural language processing tasks such as text classification, machine translation and semantic analysis. During training, the algorithm generates these vectors so that similar words have similar vectors. Since vectors are numerical representations of words, it is possible to perform mathematical operations with them, such as addition and subtraction. For example, if "man" - "woman" + "king" = "queen", we can infer that the word "queen" is related to "king" in the same way that "woman" is related to "man"[11].

Word2Vec is an algorithm composed of two learning models: the Continuous Bag of Words (CBOW), which predicts the word given its context, and the Skip-gram, which predicts the context given a word. When feeding body text into a learning model, Word2Vec generates word vectors. In this process, Word2Vec first builds a vocabulary from the training text and learns the vector representations of each word. Furthermore, Word2Vec has the ability to calculate the distance between each word. With this, similar words are grouped together based on their distances. By grouping similar words, the original feature dimension is projected to a new reduced dimension[12].

2.2.3 | Doc2Vec

Doc2vec (Document to Vector) is an extension of the Word2Vec algorithm that allows vector representation of entire documents and not just individual words. The goal of the Doc2Vec network is to learn a representation of each document such that similar documents have nearby representation vectors in vector space. Doc2Vec training is carried out using a large amount of labeled documents, where the label indicates what the main topic of the document is. Once trained, the Doc2Vec network can be used to produce vector representations of new text documents, and then can be used in various tasks such as document classification, document grouping and information retrieval[13].

2.2.4 | GloVe

GloVe (Global Vectors for Word Representation) is an embedding model that uses algebraic techniques to associate each word with a vector of numeric features. The aim is to capture the semantic and syntactic relationships between words from the context in which they appear in a large text corpus. As in Word2Vec, the vectors obtained by GloVe have the property of having semantic and syntactic similarity, for example, similar words tend to have similar vectors, and mathematical operations with the vectors can result in words related to them.

The property of semantic and syntactic similarity of the vectors obtained by GloVe is obtained from the cost function and the definition of the expected frequency of co-occurrence, which takes into account the similarity between the words. GloVe is a word embedding model widely used in natural language processing tasks such as sentiment analysis, classification and text generation[14].

2.3 | Classification Algorithms

Classification algorithms are methods used to identify new classes of observations based on training data. Algorithms learn from datasets or observations and then classify new observations into various classes or groups[15]. We present a brief review on the classification algorithms used in this manuscript below.

2.3.1 | Naive-Bayes

In probability and statistics, Bayes' theorem describes the probability of an event based on a priori knowledge that may be related to the event. Naive-Bayes classifier is a direct application of Bayes' Theorem. The term "naive" refers to the central premise of the algorithm that the considered attributes are uncorrelated with each other[16].

Bayes' Theorem expresses a relationship between new data and preexisting and known data. This algorithmic relationship between the new and the known has intuitive support in learning, since previous experience helps to interpret new acquired experiences. In line with modern philosophy, linguistics and psychology, we call this learning activity "constructivist". Bayes' Theorem offers a plausible model of this constructivist worldview, making it an important modeling tool for much of modern AI, including algorithms for natural language understanding, robotics, and machine learning[17].

The Naive-Bayes classifier is a probabilistic classification algorithm that relies on Bayes' Theorem to make predictions. It is called "naive" because it assumes that all data characteristics are independent of each other, which is not always true. However, even with this simplifying assumption, the Naive-Bayes classifier generally performs well in many classification tasks [18].

In text classification, the *Naive-Bayes* classifier is a popular method due to its computational efficiency and good performance in prediction [18]. In this classifier, each document is seen as a collection of words, where the occurrence position of each word is not considered. Because it is very simple and fast, it has a relatively higher performance than other classifiers. Also, Naive-Bayes only needs a small number of test data to complete classifications with reasonable accuracy [19].

2.3.2 | Adaptive Boosting

"*Boosting*" is a technique proposed in the 90's. The central strategy of the algorithm is to combine several inaccurate classifiers, in order to produce a new algorithm with greater precision [20].

The general concept of *boosting* may be comparated to "getting wisdom from the advice of fools" [21]. The "fools" in this case are the simple classifiers, who sort only slightly better than a coin toss. These simple classifiers by themselves are not sufficient to provide an accurate answer, but they do contain some valid information about the structure of the problem. The task of the algorithm is therefore to learn from the iterative application of a simple classifier and use that information to combine it with an accurate classification. However, calling the simple classifier multiple times on the same training data does not affect its performance. The concept of reinforcement is not manipulating the classifier itself, but rather manipulating the underlying training data by iteratively reweighting the observations. As a result, the base classifier will find a new solution on each iteration [20].

Among the various existing *boosting* implementations, *Adaptive Boosting*, or *AdaBoost* is the most commonly used [22]. The basic concept behind *AdaBoost* is to define the weights of the classifiers and train the data sample on each iteration in a way that guarantees accurate predictions of unusual observations. In each iteration, it tries to provide an optimal fit for these examples, minimizing the training error so that the next iteration makes a more accurate classification [15]. *AdaBoost* implementations typically use a Decision Tree as the base classifier, and it usually results in significantly better performance when compared to the Decision Tree alone [20].

2.3.3 | Random Forest

A Decision Tree is a sequential model that logically combines a sequence of simple tests. Each test compares a numerical attribute with a threshold value or a nominal attribute with a set of possible values [23]. With this, the algorithm is able to predict the class associated with an object traveling from the root of the tree to a leaf. At each node on the path from root to leaf, the successor child is chosen based on the division of the input space. Usually the division is based on one of the characteristics or on a predefined set of [24] rules. Decision Trees can be used to discover features and extract patterns in large databases that are important for discrimination and predictive modeling. These characteristics, along with their intuitive interpretation, are some of the reasons why decision trees are used extensively for exploratory data analysis [25].

The Decision Tree algorithm is accompanied by certain well-known limitations. These encompass the issues of instability, overfitting, reduced performance on extensive datasets, and difficulties in handling non-linear relationships. Instability refers to the sensitivity of the algorithm to even slight modifications in the training data, resulting in substantial variations in the generated decision trees and thereby compromising the overall stability of the model. Overfitting denotes a situation where the decision tree achieves a high degree of accuracy in fitting the training data but fails to effectively generalize the underlying problem to unseen data. The algorithm's performance tends to diminish when confronted with large datasets, potentially leading to computational inefficiencies. Lastly, Decision Trees encounter challenges in effectively capturing and representing non-linear relationships, posing limitations in scenarios where such relationships are prevalent [26].

In order to minimize the known limitations in the Decision Tree, the Random Forest was proposed. It is a combination of decision trees, totally independent, in such a way that each tree is built in a random subspace of the feature space. Trees in different subspaces generalize their classification in a complementary way, and their combined classification has better accuracy than the individual classification. The end result is the average or poll of predictions for every tree in the forest [27].

2.3.4 | Stochastic Gradient Descent

In mathematics, "Gradient Descent" is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the function's gradient at the current point, as this is the steepest descent direction [11].

Stochastic Gradient Descent (SGD) can be considered as a stochastic approximation of gradient descent optimization, since it replaces the real gradient, calculated from the entire dataset, by an estimate of it, calculated from from a randomly selected subset of data, thus reducing the computational load[28]. The SGD updates its parameters frequently and with a high variance, which causes the objective function to fluctuate strongly. This fluctuation allows it to eventually jump to new and potentially better local minima[29].

The SGD has the advantage of being computationally efficient, as it updates the weights after each sample, and therefore does not need to store the entire dataset in memory. In addition, it is also capable of handling large-scale datasets and can be used to train online models, i.e. models that continuously learn over time. The downside is that the SGD is prone to getting stuck in local minima, which can be mitigated through techniques like *momentum*, or using an adaptive learning rate[30].

2.3.5 | Logistic Regression

Linear models are composed of one or more independent variables that describe a relationship with a dependent response variable. One of the most commonly used linear statistical models for discriminant analysis is logistic regression[31].

Logistic regression is a statistical technique that aims to produce, from a data set, a model that allows the prediction of values taken by a categorical variable from a series of explanatory variables[32] In a problem classification, the output variable can assume only discrete values for a given set of inputs. The logistic regression model builds a regression model to predict the probability that a given data entry belongs to a category. Just as linear regression assumes that the data follows a linear function, logistic regression models the data using the sigmoid function[33].

The simplicity and interoperability of logistic regression can occasionally lead to it outperforming other sophisticated non-linear models. However, in case the response variable is extracted from a small sample, logistic regression models become insufficient and perform poorly for binary responses. In addition, logistic regression makes several assumptions, such as independence between variables and normal distribution and constant variance of responses at all levels of the explanatory variable[31].

2.3.6 | Support Vector Machine

The Support Vector Machine classifier (SVM) takes as input a set of data and predicts, for each given input, which of two possible classes the input belongs to. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or another[34].

An SVM model is a representation of examples as points in space, mapped so that examples in each category are divided by a clear space that is as wide as possible. The new examples are then mapped into the same space and predicted to belong to a category based on which side of the space they are placed[24].

SVM can use different kernel functions to map the data to a higher dimensional space, where it may be easier to find an optimal separating hyperplane. Different kernel functions also make the SVM capable of handling non-linearly separable data[35]. Some examples of common kernel functions include linear, polynomial and gaussian. Linear is the simplest kernel function, which can be used when the data is linearly separable. This kernel function performs an inner product between two input vectors, turning them into a single scalar value. Polynomial is used when the data is not linearly separable. This kernel function maps the data to a higher dimensional space by raising each feature to a specific degree. The degree of the polynomial is a model parameter that must be adjusted according to the data set. Lastly, Gaussian, also known as RBF (Radial Basis Function), is a widely used kernel function. This kernel function maps data into an infinite-dimensional space by using an exponential function that measures the distance between two input vectors. The *sigma* parameter controls the width of the exponential function and must be adjusted according to the dataset[36].

The SVM classifier is a highly effective machine learning algorithm for classification and regression problems, especially when samples cannot be linearly separated. It works by finding the separating hyperplane that maximizes the margin between samples of different classes and using support vectors to define the hyperplane. However, it can be sensitive to noise and unbalanced data, in addition to having a high computational complexity[37].

2.3.7 | Light Gradient Boosting Machine

Gradient Boosting Decision Tree is a boosting algorithm that sequentially combines multiple decision trees. Each new tree is tweaked to focus on the mistakes made by previous trees, creating a strong final model. Gradient Boosting Decision Tree is

a popular machine learning algorithm and has some effective implementations, however its efficiency and scalability are still unsatisfactory for large datasets[38].

The Light Gradient Boosting Machine (LightGBM) algorithm was developed with the proposal to overcome the limitations of the Gradient Boosting Decision Tree. The objective of the Light Gradient Boosting Machine is to reduce the number of data instances and the number of resources, thus accelerating the training process. For this, two techniques were created: Gradient-based One-Side Sampling (GOSS), which proposes to reduce the data volume by performing random sampling on instances with small gradients, while keeping all instances with large gradients, and the Exclusive Feature Bundling (EFB), which bundles unique features. With these approaches, LightGBM achieves results up to 20 times faster than its predecessor, with virtually the same accuracy[38].

LightGBM is a highly efficient machine learning algorithm designed to handle large data sets and classification and regression problems. LightGBM also supports categorical data and is capable of handling unbalanced data. It also has support for distributed parallel computing to speed up training on large datasets[39].

2.4 | Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a deep learning architecture originally built for image processing but effective for text classification. CNN's are based on traditional neural networks, but include additional processing layers, called convolution layers, which are designed to identify patterns and features in the input data, followed by pooling layers, which take each output from the map of features of the convolutional layer and prepare a map of condensed features[15].

A potential problem that arises when using CNN for text classification is the number of "channels" (feature space size). While image classification applications usually have few channels (generally only 3 RGB channels), for text classification applications this number can be very large, resulting in very high dimensionality[15].

2.4.1 | TextConvoNet

TextConvoNet is an architecture proposed by Soni, Chouhan and Rathore (2022) based on CNN for text classification. Unlike other architectures generally used for natural language processing, the TextConvoNet network takes a bidirectional matrix as input, generated by the GloVe vectorizer, and applies a multiscale convolutional operation. Figure 1 shows the architecture of the network TextConvoNet.

TextConvoNet was selected for this work because, according to the authors, it outperforms state-of-the-art machine learning and deep learning models for text classification purposes[40].

2.5 | Evaluation Metrics

Classifiers are generally trained to minimize errors. This error is evaluated using one or several metrics, the choice of which has been an ongoing debate in research and the industry for several decades[41].

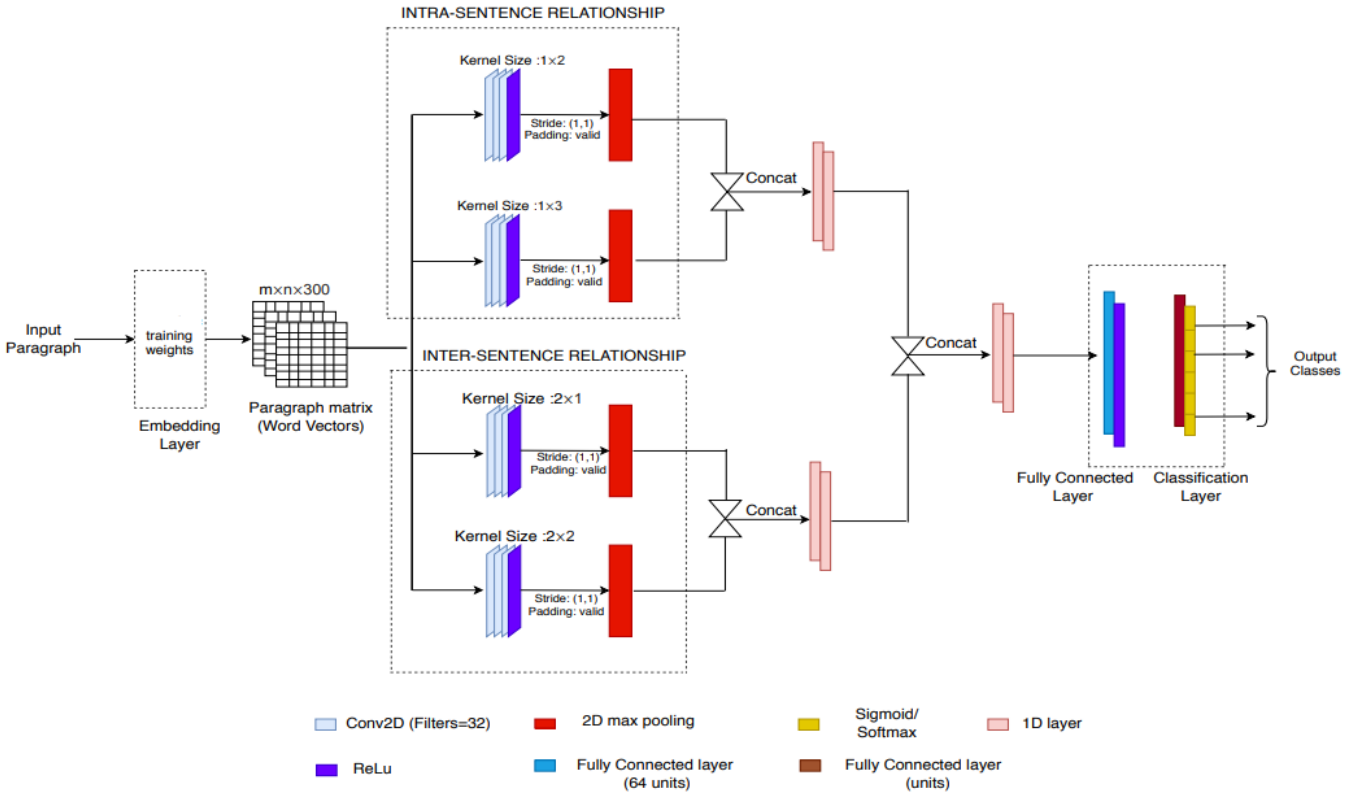
Among the metrics used in evaluating classifiers, we can mention Accuracy, Precision, Recall, F1-score, ROC, and AUC. Accuracy is defined as the proportion of true instances retrieved, both positive and negative, among all retrieved instances. Precision measures the number of correct instances retrieved divided by all instances retrieved. The Recall measures the number of correct instances retrieved divided by all correct instances. The F1-score is the harmonic mean between precision and recall[42]. ROC (Receiver Operating Characteristics Curve) plots are two-dimensional plots that display the relationship between true positive and false positive instances. Finally, the AUC (Area Under Curve) is the area under the ROC curve, a numerical representation of the same indicator[43].

2.6 | Cross-Validation

When deploying a predictive model, it is vital to understand its prediction accuracy in future tests. Therefore, the results must reflect reasonable estimates and accurate confidence intervals. Cross-validation is a widely used approach for these two tasks[44].

Cross-validation operates by splitting the dataset into two segments: one used to train and the other used to validate the model. The training and validation sets must be crossed over in successive rounds so that each data point has a chance to be validated. The basic form of cross-validation is k-fold cross-validation. First, the data is partitioned into k segments or similarly sized folds. Then k training and validation iterations are performed such that within each iteration a different fold of the data

Figure 1 TextConvoNet architecture



Source: Soni, Chouhan and Rathore[40]

is kept for validation. At the same time, the remaining $k - 1$ folds are used for learning. Upon completion, k samples of the performance metric will be available for each algorithm. Different methodologies, such as averaging, can be used to obtain an aggregate measure of these samples, or these samples can be used in a statistical hypothesis test to show that one algorithm is superior to another[45].

3 | METHODOLOGY

Initially, data was extracted from the Service Desk system. The system uses an Oracle relational database, and the extraction was performed using SQL commands. Sensitive data such as customer names, companies, users and personal information were manually removed in the extraction process. In this way, data were collected from the consultations carried out between the years 2019 and 2020, totaling 11449 tickets. Once extracted, the data were converted into a spreadsheet to allow import into the Colab development environment.

Then, a brief exploratory analysis of the data was carried out. With the visual support of a word cloud and some counters applied to the dataset, some features became evident and guided the next pre-processing step.

In the next step, data pre-processing was performed. Special characters and accents were removed and the text, originally all uppercase, was converted to lowercase. Recurrent terms with no semantic value were removed, such as greetings and closing terms (“good morning”, “yours truly” etc), in addition to the so-called “stopwords”, a set of prepositions and articles . Some recurring synonyms have also been replaced by a single term. A new analysis of dataset showed a significant improvement in word distribution.

Next, the database content was converted into numeric correspondents. First, each text was converted into a vector of words. Then, the algorithms TF-IDF and Doc2Vec were applied separately to the original dataset, generating two distinct sets of numerical data representing the documents. A preprocessed word vector database, GloVe, was also used as a requirement of the TextConvoNet architecture.

Finally, the next step consists of testing the classification algorithms. The tests were carried out in four steps: TF-IDF Vectorizer x Classic Classifiers, word embedding Doc2Vec x Classic Classifiers, TextConvoNet classifier and a proposed modification in TextConvoNet, applying as input the result of word embedding Doc2Vec. Several performance metrics were applied and summarized in graphs and tables for better comparison of results.

3.1 | Development Tools

For data processing, the Python programming language was used in the Google Collaboratory (Colab) development environment. Colab is a development platform created by Google, which allows the use of Python codes with free access to high-performance GPUs and TPUs.

Several libraries were used to implement the tests, the main ones listed below:

- *lightgbm*, which provides the LightGBM classifier,
- *sklearn*, which provides the other classic classifiers,
- *gensim*, with word embedding algorithms, and
- *tensorflow*, for the implementation of Convolutional Neural Networks.

3.2 | Dataset

The database to be used, entirely in Portuguese, was collected from a relational database of the *Conexos Help Desk* system, internal software of the company *Conexos Consultoria e Sistemas LTDA*. This system has been operating since 2004, containing a considerable collection of services the company provides to these customers. The flow of interactions in this system is similar to the flow of online forums: from a customer request, which can be a question, a problem report, or a request for an increase in the system, there may be several responses from the provider of services, as well as several replicas and rejoinders until the customer has the request resolved, at which time the service is completed. This format allows the generation of calls with complex flows, where oscillations between different subjects often occur, making their classification difficult. Thus, for tests, only calls were selected where the flow of care is composed of a question from the customer, a response from the service provider, and the end of the service. Given these conditions, data were collected from the consultations carried out between 2019 and 2020, totaling 11449 consultations. Following the Brazilian guidelines of the General Data Protection Law (*Lei Geral de Proteção de Dados - LGPD*[46]), the database was anonymized to make it impossible to identify confidential data.

Table 1 brings some statistical data on the words that make up the dataset. Among the notable characteristics, there is a strong occurrence of repetition of words in the texts.

Table 1 Dataset metrics

Indicator	Value
Total of Calls	11449
Total of Words	953728
Average of Words per Call	83.30
Distinct Words	29408
Average of Distinct Words per Call	28.90

Word Cloud is an intuitive tool that allows a quick analysis of word occurrences in a document. A Word Cloud is a visual representation of the frequency of words, so the more commonly the term appears within the text being analyzed, the larger the word will appear in the generated image[47]. Figure 2 shows the Word Cloud for the dataset.

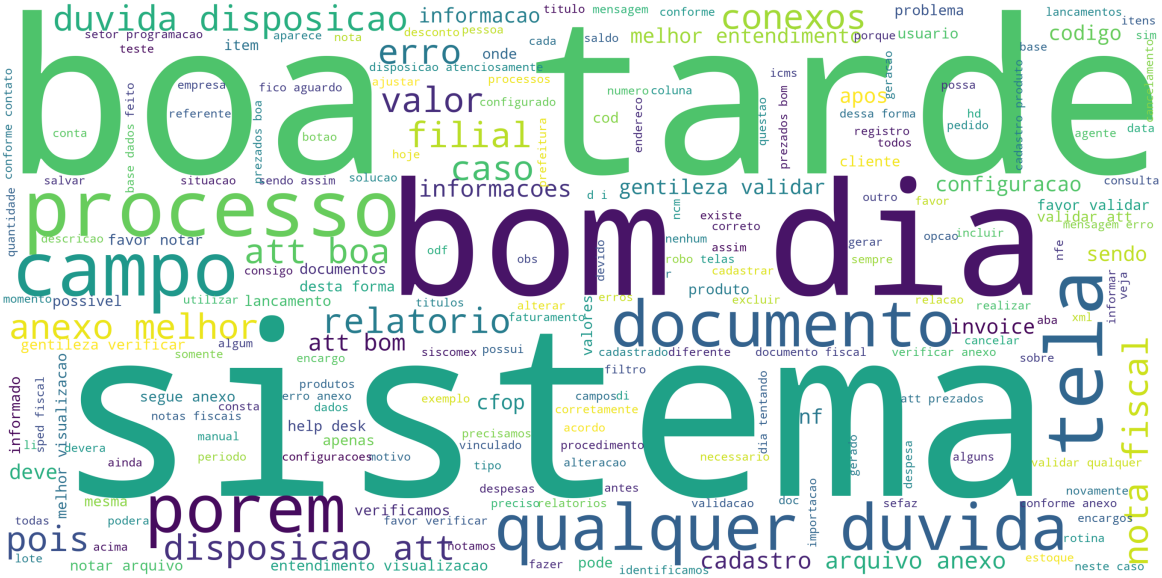


Figure 2 Dataset Word Cloud.

This Word Cloud displays an important characteristic of the dataset: the expressive presence of greetings and common terms of presentation and closing, such as "*boa tarde*" ("good afternoon"), "*bom dia*" ("good morning") and "*qualquer dúvida*" (some like "in case of doubts..."). These terms, despite representing a respectful gesture in human communication, become noise for text processing algorithms, affecting their performance[48]. Therefore, it seems reasonable to say that such terms should be removed from the dataset before classification, in order to concentrate the information and avoid possible biases in the training of the classifiers. This removal will be detailed in the next topic.

3.3 | Text Preprocessing

Text preprocessing, as an essential step in any Natural Language Processing System, involves techniques of text transformation before its analysis. It includes strategies such as removing irrelevant content for the task, clustering semantically related terms, and increasing the amount of semantic information captured[49].

In Figure 3, the Word Cloud is represented after the removal of greetings and standard introductory and closing terms. We can notice a better distribution of occurrences in terms with contextual meaning, such as the words "*processo*" ("process"), "*sistema*" ("system"), "*tela*" ("window"), and others.

3.4 | Training dataset

For training the classification algorithms, a portion of data must be previously classified, so the algorithms can be based during training and validate the obtained results. For this, a subset of 1868 attendances was classified to compose the training dataset. The unbalanced distribution between classes in the dataset makes the classifier models tend towards classes with many training samples[50]. This way, the calls were selected to compose a perfectly balanced dataset.

Table 2 shows statistical information about the subset for training. A notable feature is that, on average, relevant calls have a more significant number of words than non-relevant calls.

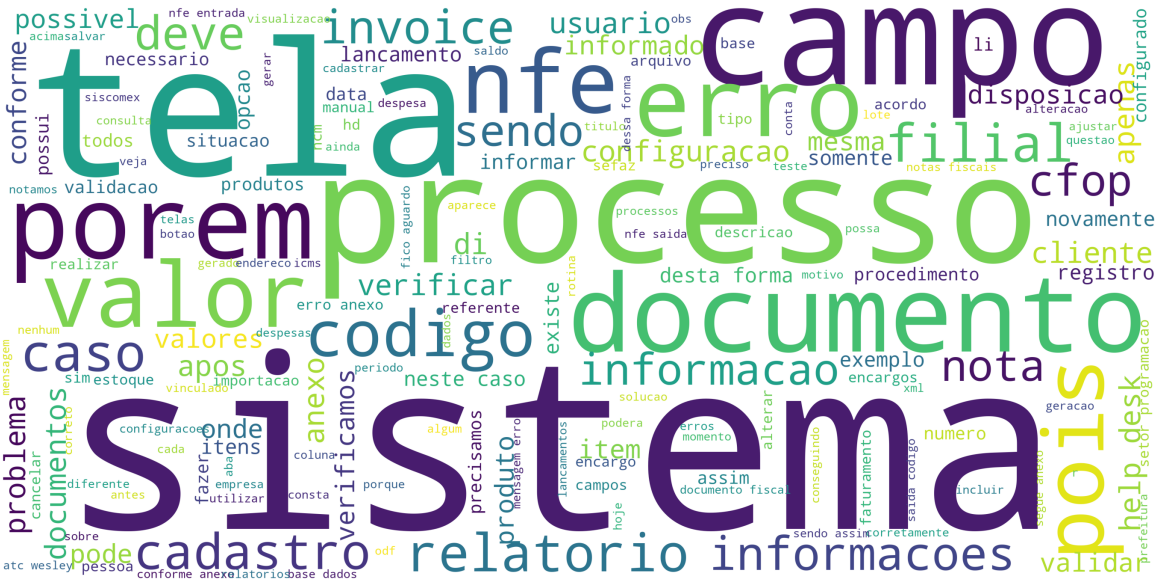


Figure 3 Dataset Word Cloud after data pre-processing.

Table 2 Descriptive information of the training dataset.

Indicator	Total	Relevants	Non-relevants
Number of Calls	1868	934	934
Total Words	88244	52347	35897
Words/Call	81.41	96.58	66.23
Distinct Words	8355	6167	5194
Distinct Words/Call	28.92	29.88	27.97

4 | RESULTS

A set of tests was carried out in an attempt to identify the concept of "relevance" of the dataset texts, divided into four scenarios:

- In the first scenario, we used TF-IDF vectorizer and Adaptive Boosting, LightGBM, Logistic Regression, Naive-Bayes, Random Forest, SGD and SVM classifiers classifiers.
- In the second scenario, we used Doc2Vec word embedding and Adaptive Boosting, LightGBM, Logistic Regression, Naive-Bayes, Random Forest, SGD and SVM classifiers.
- In the third scenario, the TextConvoNet algorithm was used, which uses the GloVe algorithm as a vectorizer, and a architecture based on CNN as a classifier.
- Finally, in the fourth scenario, a modification in the TextConvoNet algorithm was proposed, providing the vectors generated by Doc2Vec as input.

All classifiers' metrics values are obtained through cross-validation experiments with $k = 10$, performed on the training dataset.

4.1 | TF-IDF with Classical Classifiers

Next, the results using the TF-IDF algorithm and a series of classic classifiers were used in the tests, further detailed in the Table 3.

Table 3 Classic classifiers with TF-IDF when applied to the dataset.

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
AdaBoost	0.750 ± 0.028	0.743 ± 0.039	0.770 ± 0.049	0.755 ± 0.029	0.831 ± 0.030
LightGBM	0.795 ± 0.031	**0.781 ± 0.037**	0.823 ± 0.031	0.801 ± 0.029	0.876 ± 0.028
Logistic Regression	0.795 ± 0.029	0.780 ± 0.027	0.824 ± 0.045	0.801 ± 0.030	0.880 ± 0.028
Naive-Bayes	0.792 ± 0.030	0.764 ± 0.029	0.846 ± 0.041	0.803 ± 0.030	0.871 ± 0.024
Random Forest	**0.804 ± 0.028**	0.778 ± 0.032	**0.851 ± 0.027**	**0.813 ± 0.026**	**0.885 ± 0.021**
SGD	0.761 ± 0.029	0.752 ± 0.027	0.779 ± 0.045	0.765 ± 0.031	0.868 ± 0.030
SVM	0.798 ± 0.024	0.779 ± 0.029	0.835 ± 0.038	0.805 ± 0.024	0.868 ± 0.028

The experiment's classifiers presented similar metrics, emphasizing the Random Forest classifier that obtained the best results, reaching values of recall and F1-Score of 0.85 and 0.81, respectively.

4.2 | Doc2Vec with Classical Classifiers

Next, the results using the Doc2Vec algorithm and a series of classic classifiers were used in the tests, further detailed in the Table 4.

Table 4 Classic classifiers with Doc2Vec when applied to the dataset.

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
AdaBoost	0.862 ± 0.040	0.870 ± 0.043	0.852 ± 0.070	0.859 ± 0.045	0.935 ± 0.031
LightGBM	**0.922 ± 0.039**	**0.928 ± 0.027**	0.914 ± 0.056	**0.921 ± 0.042**	**0.966 ± 0.018**
Logistic Regression	0.858 ± 0.042	0.865 ± 0.037	0.848 ± 0.065	0.855 ± 0.045	0.938 ± 0.032
Naive-Bayes	0.860 ± 0.040	0.873 ± 0.065	0.850 ± 0.031	0.860 ± 0.035	0.947 ± 0.020
Random Forest	0.920 ± 0.027	0.916 ± 0.028	**0.925 ± 0.033**	0.920 ± 0.027	0.961 ± 0.023
SGD	0.822 ± 0.065	0.830 ± 0.086	0.839 ± 0.156	0.819 ± 0.087	0.934 ± 0.035
SVM	0.862 ± 0.046	0.869 ± 0.048	0.852 ± 0.057	0.860 ± 0.048	0.935 ± 0.037

A significant improvement can be seen in most indicators of all classifiers, where the classifier LightGBM and Random Forest reached metrics around 0.92. Using vectorization with Doc2Vec, the LightGBM classifier presented the best results among the used classification algorithms.

4.3 | TextConvoNet

In Table 5 we present the results of the TextConvoNet algorithm applied to the training dataset.

The results were slightly higher than the results presented by the best classical classifiers with word embedding Doc2Vec.

Table 5 TextConvoNet when applied to the dataset.

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
TextConvoNet	0.928 ± 0.017	0.925 ± 0.026	0.931 ± 0.020	0.928 ± 0.018	0.964 ± 0.013

4.4 | TextConvoNet modified

Finally, in Table 6 we present the results of a proposed modification of the TextConvoNet algorithm, providing the vectors generated by Doc2Vec as input. The motivation for this modification was to test only its core classification with the same vectorized data that were used in previous tests, for a better comparison of results. Some adjustments were necessary in the structure of the algorithm so that it received the data format provided by Doc2Vec as input.

Table 6 TextConvoNet modified when applied to the dataset.

Classifier	Accuracy	Precision	Recall	F1-Score	AUC
TextConvoNet modified	0.925 ± 0.024	0.931 ± 0.036	0.922 ± 0.037	0.925 ± 0.024	0.967 ± 0.014

The results of the proposed modification were very close to the original implementation of TextConvoNet, and to the best classic classifiers with the word embedding Doc2Vec.

4.5 | Results summary

In this topic, the main results of the tests are displayed, in order to have a better visualization of the progress in the results. Table 7 presents the metrics of the algorithms with the best results.

Table 7 Metrics of the best classifiers in this study

Classificador	Acurácia	Precisão	Recall	F1-Score	AUC
TD-IDF x LightGBM	0.795 ± 0.031	0.781 ± 0.037	0.823 ± 0.031	0.801 ± 0.029	0.876 ± 0.028
TD-IDF x Random Forest	0.804 ± 0.028	0.778 ± 0.032	0.851 ± 0.027	0.813 ± 0.026	0.885 ± 0.021
Doc2Vec x LightGBM	0.922 ± 0.039	0.928 ± 0.027	0.914 ± 0.056	0.921 ± 0.042	0.966 ± 0.018
Doc2Vec x Random Forest	0.920 ± 0.027	0.916 ± 0.028	0.925 ± 0.033	0.920 ± 0.027	0.961 ± 0.023
TextConvoNet	**0.928 ± 0.017**	0.925 ± 0.026	**0.931 ± 0.020**	**0.928 ± 0.018**	0.964 ± 0.013
TextConvoNet modified	0.925 ± 0.024	**0.931 ± 0.036**	0.922 ± 0.037	0.925 ± 0.024	**0.967 ± 0.014**

With these results, we can arrive at some definitions:

1. The word embedding algorithms produced better results than the TF-IDF vectorization algorithm. In other words, the word embedding strategies were able to maintain the relevance characteristics contained in the text during its conversion to numerical representation in a better way than the TF-IDF algorithm strategy. In fact, Exploratory Data Analysis suggested that word frequency was not a significant criterion for identifying relevance, a characteristic that is exploited by the TF-IDF algorithm.

2. The classic classifiers with the best results, regardless of the numerical representation strategy of the texts used, were LightGBM and Random Forest, both boosting algorithms based on decision trees. Decision trees are known to perform well in non-linear and/or binary classification models.

3. The best classic classifiers showed good results, with lower values, but close to the TextConvoNet network, an architecture based on Convolutional Networks.

4. The word embedding algorithms GloVe and Word2Vec presented very close results applied to algorithms based on Convolutional Networks, despite the fact that the TextConvoNet algorithm originally used a two-dimensional GloVe vector as input for multiscale analysis. This is due to the fact that Doc2Vec synthesizes the multidimensionality of Word2Vec in a unidimensional result, without major semantic losses.

5 | CONCLUSION

The definition of "relevance" of content, despite the apparent simplicity behind the binary classification, is an abstract concept. The initial exploratory analysis of the dataset showed that the characteristics that identify relevance are subtle, and involve more than just the presence or absence of words in the text.

Due to operational limitations, the training dataset was classified by only one person. The classification of relevance of the training dataset may present biases as it represents the personal opinion of those who carried out the classification. Despite this, it is important to highlight that these dataset biases do not invalidate the classification results, which proved to be effective in detecting opinion. Although a more diverse classification might be more desirable for future analyses, the results achieved in this study still provide valuable insights into the detection of opinions in limited datasets.

Several tests were carried out in an attempt to identify the concept of "relevance" of the texts in the dataset, divided into four scenarios:

- In the first scenario, the TF-IDF algorithm and Adaptive Boosting, LightGBM, Logistic Regression, Naive-Bayes, Random Forest, SGD and SVM classifiers were used as vectorizers. All classifiers presented similar metrics, close to 0.8, with emphasis on the Random Forest classifier.

- In the second scenario, the Doc2Vec vectorizer and Adaptive Boosting, LightGBM, Logistic Regression, Naive-Bayes, Random Forest, SGD and SVM classifiers were used. There was an improvement in all metrics, with emphasis on the LightGBM classifier, which presented metrics above 0.91.

- In the third scenario, the TextConvoNet algorithm was used, which uses the GloVe algorithm as a vectorizer, and a proprietary architecture based on CNN as a classifier. The metrics in this scenario fluctuated close to the value of 0.93.

- Finally, in the fourth scenario, a modification in the TextConvoNet algorithm was proposed, providing the vectors generated by Doc2Vec as input. The motivation for this modification was to test only its core classification with the same vectorized data that were used in previous tests, for a better comparison of results. Some adjustments were necessary in the structure of the algorithm so that it received the data format provided by Doc2Vec as input. The results of this modification were very close to the original implementation, close to 0.93. It is noteworthy that the objective of this work is not to validate the performance of TextConvoNet, but its applicability in the context of proposed relevance detection.

It is concluded from these results that the concept of "relevance", although abstract, is detectable both by classifiers and by complex neural networks.

The classic classifiers showed discrete results when used in conjunction with the TF-IDF vectorizer, but showed better results using the Doc2Vec embedding. That is, the Doc2Vec algorithm was able to preserve the relevance characteristic of the texts more assertively than the TF-IDF vectorizer. It is concluded that the frequency of specific words, a strategy addressed by the TF-IDF vectorizer, is not an adequate criterion for identifying relevance, a characteristic already conjectured during the Exploratory Data Analysis stage.

Additionally, these results show that the choice of a vectorizer or word embedding suitable for the problem is essential to obtain good results in classification tasks. Finally, the presented technique proved to be viable in classifying services as relevant and non-relevant in the Help Desk dataset.

As future works, a greater effort is needed in the creation of the dataset, using more human classifiers and some markup validation technique, in order to reinforce the representativeness of the concept of "relevance" by the training dataset.

References

1. Las Casas AL. *Marketing de serviços*. Atlas . 1991.
2. Costa AdSC, De Santana LC, Trigo AC. Qualidade no Atendimento ao Cliente: Um grande diferencial competitivo para as organizações. *ACM Computing Surveys (CSUR)* 2015; 02(02): 155–172.
3. Melendez Filho R. *Service Desk Corporativo - Solução com Base na Itil V3*. São Paulo: Novatec . 2011.
4. Boscolo VG. Sistema de Gerenciamento de Help-Desk. http://hdl.handle.net/11422/7469; 2009. [online, accessed 18-May-2023].
5. Lo D, Kevin Tiba K, Buciumas S, Ziller F. An Emperical Study on Application of Big Data Analytics to Automate Service Desk Business Process. In: . 2. ; 2019: 670-675
6. Cavalari GOT, Costa HAX. Modelagem e Desenvolvimento de um Sistema Help-Desk para a Prefeitura Municipal de Lavras. *Revista Eletrônica de Sistemas de Informação* 2005; 4(2). doi: 10.21529/RESI.2005.0402004
7. Baeza-Yates R, Ribeiro-Neto B. *Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca*. Editora Bookman . 2013.
8. Kang Y, Cai Z, Tan CW, Huang Q, Liu H. Natural Language Processing (NLP) in Management Research: A literature review. *Journal of Management Analytics* 2020; 7(2): 139-172.
9. Singh AK, Shashi M. Vectorization of text documents for identifying unifiable news articles. *Int. J. Adv. Comput. Sci. Appl* 2019; 10(7).
10. Rajaraman A, Ullman JD. *Data Mining*: 1–17; Cambridge University Press . 2011
11. Mikolov T, Le QV, Sutskever I. Exploiting Similarities among Languages for Machine Translation. *CoRR* 2013; abs/1309.4168. doi: 10.48550/arXiv.1309.4168
12. Ma L, Zhang Y. Using Word2Vec to process big text data. In: ; 2015: 2895-2897
13. Le QV, Mikolov T. Distributed Representations of Sentences and Documents. *CoRR* 2014; abs/1405.4053. doi: 10.48550/ARXIV.1405.4053
14. Pennington J, Socher R, Manning C. GloVe: Global Vectors for Word Representation. In: Association for Computational Linguistics; 2014; Doha, Qatar: 1532–1543
15. Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D. Text Classification Algorithms: A Survey. *Information* 2019; 10(4). doi: 10.3390/info10040150
16. Swinburne R. *Bayes's Theorem*. British Academy . 2005
17. Luger GF, Chakrabarti C. From Alan Turing to modern AI: practical solutions and an implicit epistemic stance. *AI & SOCIETY* 2017; 32(3): 321-338. doi: 10.1007/s00146-016-0646-7
18. Chen J, Huang H, Tian S, Qu Y. Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications* 2009; 36(3, Part 1): 5432-5435. doi: https://doi.org/10.1016/j.eswa.2008.06.054
19. Frank E, Bouckaert RR. Naive Bayes for Text Classification with Unbalanced Classes. In: Fürnkranz J, Scheffer T, Spiliopoulou M., eds. *Knowledge Discovery in Databases: PKDD 2006*Springer Berlin Heidelberg; 2006; Berlin, Heidelberg: 503–510.
20. Mayr A, Binder H, Gefeller O, Schmid M. The Evolution of Boosting Algorithms. *Methods Inf Med* 2018; 53(06): 419-427. doi: 10.3414/ME13-01-0122
21. Schapire RE, Freund Y. Boosting: Foundations and Algorithms. *Kybernetes* 2013; 42(1): 164-166. doi: 10.1108/03684921311295547

22. Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The Annals of Statistics* 2000; 28(2): 337 – 407. doi: 10.1214/aos/1016218223

23. Kotsiantis SB. Decision trees: a recent overview. *Artificial Intelligence Review* 2013; 39(4): 261-283. doi: 10.1007/s10462-011-9272-4

24. Shalev-Shwartz S, Ben-David S. *Understanding Machine Learning - from Theory to Algoritms*. Cambridge University Press . 2014.

25. Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD. An introduction to decision tree modeling. *Journal of Chemometrics* 2004; 18(6): 275-285. doi: https://doi.org/10.1002/cem.873

26. Rokach L, Maimon O. *Decision Trees*: 165–192; Boston, MA: Springer US . 2005

27. Ho TK. Random Decision Forests. In: . 1. ; 1995: 278-282 vol.1

28. Sra S, Nowozin S, Wright S. *Optimization for Machine Learning*. Neural information processing seriesMIT Press . 2012.

29. Ruder S. An overview of gradient descent optimization algorithms. 2016

30. Dogo EM, Afolabi OJ, Nwulu NI, Twala B, Aigbavboa CO. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. In: ; 2018: 92-99

31. Kirasich K, Smith T, Sadler B. Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. 2018.

32. Rymarczyk T, Kozłowski E, Kłosowski G, Niderla K. Logistic Regression for Machine Learning in Process Tomography. *Sensors* 2019; 19(15): 3400. doi: 10.3390/s19153400

33. Levy JJ, O'Malley AJ. Don't dismiss logistic regression: the case for sensible extraction of interactions in the era of machine learning. *BMC Medical Research Methodology* 2020; 20(1): 171. doi: https://doi.org/10.1186/s12874-020-01046-3

34. Cortes C, Vapnik V. Support-vector networks. *Machine Learning* 1995; 20(3): 273-297. doi: https://doi.org/10.1007/BF00994018

35. Burges CJ. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery* 1998; 2(2): 121-167. doi: 10.1023/A:1009715923555

36. Ting KM, Zhu Y, Zhou ZH. Isolation Kernel and Its Effect on SVM. In: KDD '18. Association for Computing Machinery; 2018; New York, NY, USA: 2329–2337

37. Bhavsar H, Panchal MH. A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 2012; 1(10): 185–189.

38. Ke G, Meng Q, Finley T, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: Guyon I, Luxburg UV, Bengio S, et al., eds. *Advances in Neural Information Processing Systems*. 30. Curran Associates, Inc.; 2017.

39. Machado MR, Karray S, Sousa dIT. LightGBM: an Effective Decision Tree Gradient Boosting Method to Predict Customer Loyalty in the Finance Industry. In: ; 2019: 1111-1116

40. Soni S, Chouhan SS, Rathore SS. TextConvoNet: A Convolutional Neural Network based Architecture for Text Classification. 2022. doi: 10.48550/ARXIV.2203.05173

41. Yacouby R, Axman D. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: ; 2020: 79–91.

42. Dalianis H. Evaluation metrics and evaluation. In: Springer. 2018 (pp. 45–53).

43. Zhang X, Li X, Feng Y, Liu Z. The use of ROC and AUC in the validation of objective image fusion evaluation metrics. *Signal processing* 2015; 115: 38–48. doi: 10.1016/j.sigpro.2015.03.007

44. Bates S, Hastie T, Tibshirani R. Cross-validation: what does it estimate and how well does it do it?. *arXiv preprint arXiv:2104.00673* 2021.

45. Refaeilzadeh P, Tang L, Liu H. *Cross-Validation*: 1–7; New York, NY: Springer New York . 2016

46. Brasil . Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). *Diário Oficial da República Federativa do Brasil* 2018.

47. Atenstaedt R. Word cloud analysis of the BJGP. *British Journal of General Practice* 2012; 62(596): 148–148. doi: 10.3399/bjgp12X630142

48. Hajrizi R, Nuçi KP. Aspect-Based Sentiment Analysis in Education Domain. *CoRR* 2020; abs/2010.01429.

49. Hickman L, Thapa S, Tay L, Cao M, Srinivasan P. Text Preprocessing for Text Mining in Organizational Research: Review and Recommendations. *Organizational Research Methods* 2022; 25(1): 114-146. doi: 10.1177/1094428120971683

50. Okkalioglu M, Okkalioglu BD. AFE-MERT: Imbalanced Text Classification with Abstract Feature Extraction. *Applied Intelligence* 2022: 1–17. doi: 10.1007/s10489-021-02983-2

How to cite this article: Degasperi, M. M., Cavalieri, D. C., and Castro, F. Z. (2023), Relevance classification on service desk texts using Natural Language Processing, *Wiley Periodicals LLC*, *2023*.