Classification of Requirements Ambiguity Through Machine Learning Techniques

Carla D. M. Vieira[1]

[1]Institute of Exact Sciences and Informatics (ICEI)
Pontifical Catholic University of Minas Gerais (PUC Minas)
Belo Horizonte, MG - Brazil

camvieira@sga.pucminas.br

Abstract. *Requirements Engineering is a fundamental step for software development, moreover the developed requirements quality directly influences the development success. Generally, the quality assessment of the requirements is done manually by the engineers themselves, making the process slower, subjective and susceptible to errors. As an alternative to this evaluation, Machine Learning algorithms can be used to evaluate the requirements quality written in natural language in an automated way. This paper proposes a comparative analysis between the combination of two vectorization techniques (Bag of Words and Term Frequency-Inverse Document Frequency) and four supervised learning algorithms (k-Nearest Neighbor, Support Vector Machines, Logistic Regression and Multinomial Naive Bayes) for the evaluation of the unambiguous quality characteristic of requirements. For each algorithm, their performance values are analyzed and compared with each other. Considering the mean metrics as F1-Score, the Logistic Regression showed the best performance, followed by Support Vector Machines and Multinomial Naive Bayes. For future work involves the possibility of using more advanced machine learning tools as semantic tagging, to archive better results.*

Keywords— *Requirements Engineering, Requirements Quality, Machine Learning, natural language, ambiguity*

Bachelor in Software Engineering - PUC Minas
Trabalho de Conclusão de Curso (TCC) - Undergraduate Thesis

Content advisor (TCC I): Laerte Xavier - laertexavier@pucminas.br
Academic advisor (TCC I): Lesandro Ponciano - lesandrop@pucminas.br
Advisor (TCC II): Hugo Bastos de Paula - hugo@pucminas.br

Belo Horizonte, May 15, 2022.

1. Introduction

Requirements Engineering is an important interdisciplinary function of Software Engineering, being responsible for identifying the needs of stakeholders [Souza et al. 2020] and also define and maintain the requirements that the desired system, software, or service must fulfill [ISO/IEC 2018]. This field has a fundamental role for quality assurance in software development since having well-defined requirements is an important starting

point to it. In parallel, the integration of Artificial Intelligence technologies into Software Engineering aims to optimize the process of developing software products, automate costly tasks and to create systems with high quality [Gramajo et al. 2020b]. This way, the use of Artificial Intelligence presents itself as an alternative to the evolution of the qualification of requirements in an objective and automated way.

As requirements are written in natural language, the assessment of their quality is usually limited to the knowledge and experience of the engineers responsible, which makes it difficult to standardize and quickly analyze the quality of the requirements. Currently, the possibility of automating this process is being studied through the different Machine Learning techniques available, but it is not clear which method may be more effective for this purpose. The different Machine Learning algorithms can present different results for the performance metrics of the models, and it is necessary to understand which one would best suit the context of classification of requirements by quality. Therefore, the problem that this paper seeks to solve is the **lack of empirical analysis for the use of algorithms for automatic classification of software requirements written in natural language in different quality classes of requirements using Machine Learning techniques**.

The process of defining and validating requirements consumes a lot of time and effort [Gramajo et al. 2020a], highlighting the importance of the improvement of this stage of software development. Furthermore, as requirements are written in natural language, the flexibility and inherent features of the language, such as inconsistencies, redundancies, and ambiguities, can lead to errors during specification and, consequently, can negatively influence later phases of the product life-cycle [Gramajo et al. 2020b]. Added to the importance that the Requirements Engineering stage has for the quality assurance of the developed software, the points raised guarantee the need to improve the requirements quality assessment practices, in a more objective and automated way or the best possible performance.

In order to analyze expressions in natural language, it will be needed to use the two best-known vectorization techniques, "Bag of Words" (BoW) and Term Frequency—Inverse Document Frequency (TF -IDF). These techniques are analyzed in combination with four supervised learning algorithms, which are: k-Nearest Neighbor (k-NN), Support Vector Machines (SVM), Logistic Regression (LR), Multinomial Naive Bayes (MNB).

Thus, this work aims to **evaluate the use of automatic classification of software requirements ambiguity, using Machine Learning techniques**. For this, the three specific objectives of this work are: 1) classify a requirements database by ambiguity; 2) train the automatic classification models, and 3) analyze the performance metrics of the models, comparing them with each other.

The result of this work will is a comparative analysis between the different models to classify the requirements by quality, presenting their performance difference in each possible combination of feature extraction and Machine Learning algorithm. For this, performance metrics such as accuracy, precision, recall and *F1-Score* are evaluated.

This work is divided as follows: Section 2 presents the theoretical foundations. Section 3 addresses work related to requirements quality classification using machine

learning techniques. Section 4 reports the methodology and materials used, followed by Section 5, with the presentation of the results. Section 6 presents the discussion around what was found and the threats to validity. Finally, Section 7 presents the final conclusion and outlook for future works.

2. Theoretical foundation

In this section, the main concepts and techniques that are involved in the solution of the presented problem are detailed. They are: Requirements Engineering, Artificial Intelligence, Machine Learning and supervised learning and techniques, algorithms and metrics used in the development of the study.

2.1. Requirements Engineering

Requirements Engineering (RE) is the area of Software Engineering that provides the appropriate mechanisms to understand what the customer wants, analyze needs, assess feasibility, negotiate a reasonable solution, specify the solution, validate the specification and manage the requirements as they are transformed into a working system [Thayer et al. 1997]. The activities of this process include: obtaining, analyzing, specifying, validating, verifying and managing requirements [Pohl 2010].

Requirements are defined by natural language sentences and to guarantee the quality of an individual requirement it is necessary that it follows the following characteristics: necessary, free implementation, unambiguous, consistent, complete, unique, feasible, traceable and verifiable.

2.1.1. Ambiguity in Requirements Specification

Ambiguity on daily bases is defined as the capability of being understood in two or more possible senses or ways. The IEEE (Institute of Electrical and Electronics Engineers) Recommended Practice for Software Requirements Specifications defines that ”A Software Requirement is unambiguous if, and only if, every requirement stated therein has only one interpretation.” Presumably, a requirement is ambiguous if it is not unambiguous. Despite being a first established definition for the definition of ambiguity, it is still a vague statement so we can move on to the concrete classification of a requirement as ambiguous or not.

To have a better structure on how to identify ambiguity in requirements, we used [do Prado Leite and Doorn 2004] definition that divides the ambiguities found in requirements into linguistic ambiguity, Software Engineering ambiguity, vagueness and generality, as shown in the taxonomy tree for ambiguity in Figure 1.

Linguistic ambiguity, can be distinguished into three broad classes: lexical ambiguity, syntactic ambiguity and semantic ambiguity.

Lexical ambiguity occurs when a word has several meanings. Syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning. Semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or structural ambiguity [do Prado Leite and Doorn 2004].

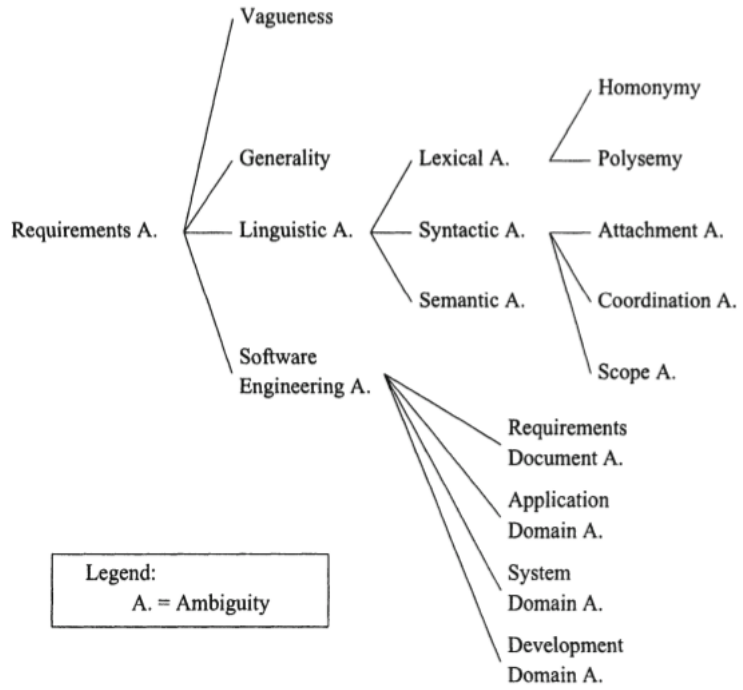

Figure 1. Requirement Ambiguity Taxonomy Tree

Vagueness and generality, also called indeterminacy, are closely related to what s known as ambiguity. [do Prado Leite and Doorn 2004] distinguishes a general word or phrase from an ambiguous word or phrase, each is open to more than one interpretation.

Software Engineering ambiguity occurs when there is more than one interpretation taking into account the sources you can have in different domains, as documentation, application, system and development.

2.2. Artificial Intelligence

The term Artificial Intelligence can be considered a relatively broad concept, considering that there is no final consensus on its meaning, collecting a set of different definitions. One of the most used definitions, although not universally accepted, is that AI is the study of how to make computers perform tasks that people are currently better at [Rich 1985].

The different aspects related to AI are defined as studies of ways to establish "intelligent" behaviors in machines. With the emergence of new learning strategies and the increase in the processing power of computers, AI has become a more accessible tool for the market and for improving the development of software systems[Feldt et al. 2018].

2.3. Machine Learning

Machine Learning (ML) is an AI discipline composed of a set of techniques that allow computers to learn data and make generalizations and predictions from them, which facilitates decision making [Gramajo et al. 2020a]. Although not a new concept, the availability of large volumes of data and greater processing power in computers has allowed us to experiment and investigate further its uses and applications in various domains.

One of the first definitions of the ML concept was the field study that gives computers the ability to learn something, for which they were not explicitly programmed [Samuel 1959]. A more formal definition is that expressed by Mitchell (1997), which states that a computer learns a specific task T, considering type E experiences, concerning a performance measure P, if the computer improves its performance P, in the task T, from experience E.

Machine learning can be classified as supervised or unsupervised. It is considered supervised learning if the data used is previously labeled, classified, or categorized. Otherwise, learning is considered unsupervised. In this work, supervised learning is used.

2.4. Supervised Learning

Supervised learning is an ML technique that uses labeled data, called training data, to build a predictive model to predict the label of unlabeled data [Gramajo et al. 2020a]. The training data set includes input and response value data. From this, the supervised learning algorithm seeks to create a model that can make predictions about the response values for a new data set. A test data set is usually used to validate the model. If larger training data sets are used, it is possible to generate models whose predictive capacity is greater, obtaining good results in new data sets [Darnstädt et al. 2014].

2.5. Techniques and Algorithms

In this subsection, the concepts of techniques and algorithms that will be used in the development of the study are presented.

2.5.1. Resource Extraction

Machine learning algorithms operate in a numeric feature space, expecting input as a two-dimensional *array* where rows are instances and columns are featured [Canedo and Mendes 2020]. To perform machine learning in text, as is the case of the analyzed requirements, it is necessary to transform the sentences into vector representations for the application of numerical machine learning. The process of encoding documents into a numeric feature format is called feature extraction or, more simply, vectorization and is an essential first step in natural language analysis [Canedo and Mendes 2020].

To deal with the natural language in the requirements, two alternatives for resource extraction are the "Bag of Words" (BoW) and "Term Frequency—Inverse Document Frequency in Documents" (TF -IDF).

BoW: is a way of representing the text according to the occurrence of words in it. The definition of "Bag of Words" is related to not taking into account the order or structure of words in the text, only whether they appear or how often they appear.

TF-IDF: uses statistical measures to measure how important a word is in a document (text) through a score. The TF-IDF of a word in a text is done by multiplying two different metrics: the *Term Frequency* (TF), which measures how often a term occurs in a document, and the *Inverse Document Frequency* (IDF), which measures how important a term is in the context of all documents.

2.5.2. Machine Learning Algorithms

In this work, we compare the use of four different supervised learning algorithms, k-Nearest Neighbor (k-NN), Support Vector Machines (SVM), Logistic Regression (LR) and Multinomial Naive Bayes (MNB):

k-NN: is based on the principle that instances within a dataset will generally exist in close proximity to other instances that have similar properties. The algorithm classifies new data by calculating the distance of this data with the existing instances within the database. It then selects the k closest instances and calculates their mean, for regression problems, or obtains the mode.

SVM: is a machine learning model capable of performing linear or non-linear classification, regression, and even detection of *outliers*. The algorithm does the classification by creating a maximum-margin linear hyperplane that separates two classes. This margin means that there is little possibility of separating the sample data and, therefore, there is little chance of misclassification of new instances.

LR: Logistic regression is commonly used to estimate the probability of an instance belonging to a specific class. Abdul et al. (2019) define logistic regression as a class of regression in which an independent variable is used to predict the dependent variable. It is called binary logistic regression when the dependent variable has two classifications. It is called multinomial logistic regression when the dependent variable has more than two classes. In this work, we used binary logistic regression to classify requirements as ambiguous or unambiguous.

MNB: is a generative model that estimates the conditional probability of a class with input data. In particular, *Naive Bayes* assumes that the input resources are independent of each other (conditional independence). Multinomial Naive Bayes is a specialized version of Naive Bayes used primarily for document and text classification.

2.6. Metrics

Evaluation metrics are primarily used to evaluate the performance of a classifier. The performance is verified through mathematical formulas that will compare the predictions obtained by a model with the real values of the database. The metrics analyzed for the purpose of comparing the implemented methods will be the precision equation, the recall equation, and the *F1-Score* equation. For a better understanding of the definitions of these metrics, Table 1 is taken into account, which indicates the errors and successes of the model, compared with the expected result, called the confusion matrix. In it, the concepts of True Positive, True Negative, False Positive, and False Negative are defined according to the classification response made and the actual response in the database.

Table 1. Confusion Matrix

		Classification	
		Yes	No
Real	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative (TN)

2.6.1. Accuracy Equation

Accuracy is a general metric to describe how the model performs across all classes, so it is useful when all classes are of equal importance. Regarding the Confusion Matrix shown in Table 1, the Equation 1 defines how the accuracy is calculated as the ratio between the number of all correct predictions to the total number of predictions from all classes. As in the context presented, the difference between the classes needs to be taken into consideration. Thereat, the other metrics can be calculated specifically for each class, which can be of great interest for this study.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2.6.2. Precision Equation

Precision measures the percentage between the number of correctly classified positive samples in relation to the total positive samples. Precision is calculated using the ratio of the total positive ratings correct to the total positive ratings performed for the class. This calculation can also be seen as the ratio of the number of true positives (TP) to the number of positives detected by the model (TP + FP), as defined in Equation 2.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

A precision of 1 indicates a percentage of 100% of the classifier and that all instances were classified correctly.

2.6.3. Recall Equation

The *Recall* is a proportion of positive instances that are correctly detected by the classifier. As defined in Equation 3, it is calculated as the number of times a positive class was correctly predicted (TP), divided by the number of times the class appears in the test data (TP + FN). In other words, it calculates among all situations of positive class as expected value, how many are correct.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

2.6.4. F1-Score Equation

The *F1-Score* is considered the harmonic mean between precision and recall, taking into account both metrics simultaneously and its calculation is defined in Equation 4. While regular mean treats all values equally, harmonic mean gives much more weight to low values. As a result, the classifier will only get a high *F1-Score* if recall and precision are high [Canedo and Mendes 2020].

$$F1 - Score = \frac{2 * (Accuracy * Recall)}{Accuracy + Recall} \quad (4)$$

3. Related Works

In this section, studies that present proposals related to the use of machine learning for the evaluation of requirements quality are discussed. The first article presents a comparative analysis with the same combinations of techniques evaluated in this work, but for classification between types of requirements. This is followed by an article that explores a complete approach to automatic requirements improvement assessment and suggestion. Subsequently, the following two articles explore the analysis of quality characteristics, namely uncertainty and uniqueness, respectively. Finally, the following studies present a study using Bayesian Networks and, finally, the identification of harmful ambiguities.

Dias et al. (2020) present a very similar proposal regarding the comparative analysis of Machine Learning techniques for better performance in requirements classification. The article presents the same combination of two vectorization techniques (BoW and TF-IDF) and four supervised learning algorithms (k-NN, SVM, LR, and MNB) for performance analysis, however, it aims to classify the requirements between Requirements Functional and Non-Functional and sub-classify this second group. The best result found was the use of the TF-IDF followed by the use of the LR. The differential factor of this article in relation to Dias et al. (2020) is the focus on quality classification, more precisely on the unambiguity of requirements.

Having the highest possible quality set of requirements is of great importance and the benefits include improving project quality, better understanding customer needs, reducing costs, and predicting project schedules and outcomes more accurately [Adanza Dopazo et al. 2021]. Thus, the main objective of this article is to create a methodology that can modify poor requirements, extracting the main characteristics of each requirement, evaluating its quality with a high level of expertise, and then improving the quality of the requirements. In the first step, a machine learning algorithm was implemented to classify requirements based on quality and identify those that are most likely to be problematic. And in the second stage, the genetic algorithm generated solutions to improve the quality of the requirements identified as inferior. The genetic algorithm obtained results that can be compared with the theoretically optimal solution, defining this tool as promising for requirements engineering.

Stakeholders often use speculative language when they need to convey their requirements with some degree of uncertainty [Yang et al. 2012]. Due to the intrinsic imprecision of speculative language, speculative requirements are at risk of being misinterpreted and the related uncertainty overlooked. Thus, they can benefit from careful handling in the requirements engineering process. In this article, a linguistically oriented approach for automatic detection of uncertainty in Natural Language (NL) requirements is presented. First, speculative sentences are identified by applying a machine learning algorithm called Conditional Random Fields (CRFs, *Conditional Random Fields*) to identify uncertainty clues. The algorithm exploits a set of lexical and syntactic features extracted from requirement sentences. Second, the approach attempts to determine the scope of uncertainty. A rule-based approach is used that relies on a set of hand-made linguistic

heuristics to determine the scope of uncertainty with the help of dependency structures present in the sentence parsing tree. The results seem promising for the implementation of uncertainty assessment of real requirements. Like this work, Yang et al. (2012) study the field of automatic analysis of requirements uncertainty, however, they only evaluate the use of the CRFs algorithm and do not intend to perform a comparative analysis between the algorithms.

One of the challenges in evaluating the quality of requirements is dealing with a text written in natural language [Gramajo et al. 2020b]. In this article, Gramajo et al. (2020) propose another study of the quality of requirements developed through neural networks. Using the definition of quality defined by the ISO/IEC/IEEE 29148:2018 standard, the study focuses on one of the quality topics of the requirement, the uniqueness. For this, it is proposed to use neural networks, of the type Long Short-Term Memory (LSTM), to predict whether the requirements are singular or not. Each requirement is subjected to a process of tokenization and grammatical labeling in order to obtain a representative sequence that acts as an input to the neural model. And then the proposed neural model is trained with a dataset of 1000 requirements, 80% for training the neural network, 10% for testing, and 10% for validation. The results presented by the model are defined as promising and the interest in continuing the research with the other quality topics of a requirement stands out. Gramajo et al. (2020) studies the evaluation of the characteristic of requirements defined as uniqueness, while this work complements it with the study of the characteristic of unambiguity.

The Requirements Engineering area is crucial for the development of software that does not depart from the customer's needs but is often hostage to the expertise and experience of the responsible professional [Wiesweg et al. 2020]. This is touted as an opening for failure and therefore risk. With this, Wiesweg et al. (2020), in an attempt to help this risk management, developed a solution using Bayesian Networks. For this, different versions of Bayesian Networks that model the relationships between causes, problems, and effects in RE were evaluated. The models were trained with data that were collected through two surveys with responses from 228 and 488 professionals, about problems, causes, and effects found in real projects. The models were intended to perform a *post-mortem* analysis (given the problem, diagnose the cause) and predictive (given the cause, diagnose possible problems). Both models presented good values of *recall* and precision, demonstrating a possible opening for use in real risk management. For this article, the work by Wiesweg et al. (2020) is used as a reference in the use of machine learning to analyze the consequences of low-quality requirements for software development.

In another article, Yang et al. (2010) propose to automatically identify potentially harmful ambiguities of requirements, which occur when the text is interpreted differently by different readers. For this, a set of ambiguities was extracted along with several human judgments about their interpretations. The judgment distribution was used to determine whether the ambiguity is harmful or innocuous. Using machine learning techniques, an automated tool was developed to predict the antecedent preference of noun phrase candidates, which in turn is used to identify harmful ambiguities. The study showed results that allow exploring and highlighting realistic and potentially problematic ambiguities in real requirements documents. This article presents an in-depth study on the identification of harmfulness ambiguities of requirements, a characteristic to be evaluated in the

algorithms used in this article.

4. Materials and Methods

This work is classified as quantitative and explorative research since it aims to evaluate and bring a comparative analysis between Machine Learning techniques in the automatic quality classification of software requirements expressed in natural language. Thus, using two feature extraction methods and four different machine learning algorithms are used in the context of requirements quality classification. This section presents the materials, methods, evaluation metrics, and research execution stages.

4.1. Materials

The database used is PROMISE_exp [Lima et al. 2019] available at `https://tinyurl.com/PROMISE_exp`. PROMISE_exp is an expanded version of the original PROMISE suite, a public repository inspired by the *UCI Machine Learning Repository*, and created to encourage repeatable, verifiable, falsifiable, and/or improbable predictive software engineering models. The original repository consists of a pre-labeled set in classification (Functional and Non-Functional Requirements) of 255 functional and 370 non-functional requirements. The expanded version, used in this paper, consists in 985 requirements label with the sub-category of non-functional requirement such as availability and usability, among others. In order to use this database, it is necessary to pre-label these requirements according to the value of the quality characteristic to be evaluated, in the case of this paper, unambiguity.

4.2. Methods and Metrics

In the Section 2, related to the theoretical foundation, the concepts of the techniques, supervised machine learning algorithms and equations of the metrics used in the development of the research are presented.

The requirements are classified into two classes, namely ”Ambiguous” and ”Unambiguous”. Following the definition of identifying ambiguity defined in Subsection 2.1.1, the requirement was classified as ambiguous if it was identified at least one of the type of ambiguities: Vagueness/Generality, Linguistic Ambiguity and Software Engineering Ambiguity.

Before the feature extraction, the requirements go through a process of normalization. All stop-words such as pronouns and articles are removed and with the use of a Stremmer, which inflected verbs and nouns are converted to their original form. The Snowball Stemmer was used as it works well for the English language.

For feature extraction, the two best-known techniques are used, BoW and TF-HDI, defined earlier in Section 2.5.1. The database is then separated into 70% for the training and 30% for testing. In order to mitigate the problem faced by the training database imbalance, the Synthetic Minority Oversampling Technique (SMOTE) strategy is used, which basically involves duplicating examples in the minority class, in this case, ambiguous requirements.

Regarding supervised machine learning algorithms, k-NN, SVM, LR, and MNB are used. For each combination between a resource extraction technique and an algorithm,

the labeled database is used in order to promote a supervised learning process. The feature extraction technique will be responsible for the process of encoding the requirement in a numerical format and then the algorithm is fed by pairs of known inputs and outputs, subsequently, in the form of vectors. For each result found, the accuracy, precision, *recall* and *F1-Score* are used for the comparative analysis of the performance of each one. For this final step before the result analysis, it is crucial for our goal to have these metrics for reach class, the general average and the weighted average, so there is a bigger picture if how each performance scored.

4.3. Execution Steps

For the execution of the work, the following steps are considered:

1. Collection of requirements database;
2. Classification of database requirements by quality characteristic and database increment. The base will be classified between ambiguous or unambiguous requirements;
3. Normalization: data cleaning, where all stop-words such as pronouns and articles are removed. Inflected verbs and nouns are converted to their original form with the stremmer;
4. Resource Extraction: the transformation of text into numerical information. In this study, BoW and TF-IDF are used to perform the conversion;
5. Split Test and Train sets: 70% of the data is allocated to the training and 30% for testing;
6. Balancing training sample: as ambiguous class considerably smaller then unambiguous, an oversampling strategy is used;
7. Training: vectors obtained in the Feature Extraction phase are used to train and predict the classification models of the four algorithms used in this paper: SVM, MNB, k-NN, and LR;
8. Results: results of requirements label predictions and the actual labels of those requirements are used to calculate the performance measures described in Section 2.6.

5. Results

During this section, the results found from the methodology developed will be discussed with the metrics previously defined. After performing the database classification, 89% of the requirements were classified as unambiguous and 11% ambiguous. Due to this imbalance, it is necessary to use the SMOTE, oversampling technique. The coding process and the metrics results can be found at this Jupyter Notebook.

5.1. Accuracy

The accuracy of the four analyzed models already indicates a superior performance of Support Vector Machine and Logistic Regression, both with 91%, in relation to the others, highlighting the worst result found for K-Nearest Neighbor, which was the only one that presented a result below 80%.

Table 2. Accuracy values

	Accuracy
Naive Bayes	85%
K-Nearest Neighbor	51%
Support Vector Machine	91%
Logistic Regression	91%

5.2. Precision

Previously defined by the Equation 2, precision allows us to understand deeper, of the requirements classified as ambiguous, which is the percentage classified right. The k-NN model had a very high precision for the unambiguous requirements, that is, all requirements classified as unambiguous were actually unambiguous. However, the result of 18% precision for ambiguous is a considerably lower than the others, which indicates that the model had a bigger tendency to classify the requirements as ambiguous.

The unambiguous class precision was good enough in all models, however, it is possible to notice that there is a discrepancy in the values found for the ambiguous class, of greater interest for this paper. The only one that presents a considered good precision (above 80%) was the Support Vector Machine.

Table 3. Precision values

	Precision			
	Unambiguous	Ambiguous	Macro Avg	Weighted Avg
Naive Bayes	0.95	0.37	0.66	0.89
K-Nearest Neighbor	1.00	0.18	0.59	0.91
Support Vector Machine	0.91	0.83	0.87	0.90
Logistic Regression	0.94	0.58	0.76	0.90

5.3. Recall

The recall mostly indicates the percentage of right positive instances that were correctly classified from all the instances in the original class, as exemplified in Equation 3. Analyzing the Table 4, LR and SVM tend to have a good recall for unambiguous requirements, which means all the unambiguous requirements in the test sample were correctly identified. This also happens for ambiguous requirements in K-NN. However, for the unambiguous class in K-NN and ambiguous class in LR and SVM the result are below 50%. This can mean that the K-NN model had a tendency to classify more requirements originally unambiguous as ambiguous and the models LR and SVM classify ambiguous requirements as unambiguous.

Table 4. Recall values

	Recall			
	Unambiguous	Ambiguous	Macro Avg	Weighted Avg
Naive Bayes	0.88	0.58	0.73	0.85
K-Nearest Neighbor	0.45	1.00	0.72	0.51
Support Vector Machine	1.00	0.16	0.58	0.91
Logistic Regression	0.96	0.48	0.72	0.91

5.4. F1-Score

Knowing that mostly the precision improvement comes with the recall decreasing, the F1-score can give us the harmonic average of precision and recall, showing an impression of the general balance between the two metrics. In Table 5, MNV, k-NN and LR achieved values above 90% for the unambiguous class. However in the ambiguous class, of greater interest, the only model to reach a satisfactory result of more than 50% was the LR. As a result, the macro and weighted average from LR were better than the others.

Table 5. F1-Score values

	F1-Score			
	Unambiguous	Ambiguous	Macro Avg	Weighted Avg
Naive Bayes	0.91	0.45	0.68	0.87
K-Nearest Neighbor	0.62	0.30	0.46	0.59
Support Vector Machine	0.95	0.27	0.61	0.88
Logistic Regression	0.95	0.53	0.74	0.91

6. Discussion and Threats to Validity

As a general analysis of the performance of the models, the Logistic Regression was the model that presented the most satisfactory results among the four, followed by the Support Vector Machine and Naive Bayes. This happens because it presented the best macro and weighted F1-Score average, with a good unambiguous classification and a considerable satisfying classification of ambiguous requirements. However, the selection of the best model could vary from the interest in the choice, as for example, the K-Nearest Neighbor could be a better choice when the intention is to identify as much as possible from the ambiguous requirements, even if this occurs as a result of more wrong classification of unambiguous requirements as ambiguous, with what we call False Positives .

Although were found relatively acceptable values for some models, the metrics values would not be sufficient for implementation and market acceptance. In this sense, an important point to take into account about the study presented is that the definition of ambiguity is often related to the entire set of grammatical elements relating to each other. To deal with the obstacle that is the identification of ambiguity in natural language, studies are opting for the use of more robust machine learning technologies than the classic models, such as the semantic tagging, that is, tagging each word on your semantic type and analyze the relationship between them, which did not fit the scope of this study.

And another point for the threats to validity for this paper, it is important to note that the database classification was made by only one person. Even following a classification methodology, the ideal to avoid subjectivity in the classification will be to have at least three different classifiers, as it will be able to choose a definitive answer in case of a tie.

Regarding the distribution of the database, the imbalance between the class of ambiguous requirements and the class of unambiguous ones that can be considered a threat to validity, since it was necessary to use strategies such as over sample.

7. Conclusion and Future Work

In this paper, a comparative analysis was presented on the use of four different machine learning models for the classification of ambiguity in software requirements written in natural language.

For this, a database with real requirements was used, maintained for use by the community for predictive models of software engineering, the PROMISE_exp. From this database, a manual classification was made between ambiguous and unambiguous requirements, considering the five sub-types of ambiguities that can be found in software engineering requirements. With the use of Natural Language Processing tools, like features extraction and stemming, the data was prepared for the four Machine Learning algorithms selected for the analysis, k-Nearest Neighbor, Support Vector Machines, Logistic Regression and Multinomial Naive Bayes.

Based on a more in-depth analysis of the metrics of precision, recall and F1-score of each class, ambiguous and unambiguous, it identified that Logistic Regression might obtain better results for the context proposed, followed by Support Vector Machine and Naive Bayes and for last, K-Nearest Neighbor. However, it is important to emphasize that these are only preliminary results, with values that would need to be refined to archive satisfactory measures for use in the market, like different perspectives of classifications.

For potential future work, it is noteworthy the exploration of more complex classification algorithms, taking into account the tagging of words semantically. This type of procedure have a high potential for have better results in the context as the identification of ambiguity is commonly related to the placement of other words semantically.

Replication Pack

The replication package for this work is available at:

`https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2021-2-tcci-5308100-pes-carla-vieira`

References

[Adanza Dopazo et al. 2021] Adanza Dopazo, D., Moreno Pelayo, V., and Génova Fuster, G. (2021). An automatic methodology for the quality enhancement of requirements using genetic algorithms. *Information and Software Technology*, 140:106696.

[Canedo and Mendes 2020] Canedo, E. D. and Mendes, B. C. (2020). Software requirements classification using machine learning algorithms. *Entropy*, 22(9).

[Darnstädt et al. 2014] Darnstädt, M., Simon, H. U., and Szörényi, B. (2014). Supervised learning and co-training. *Theoretical Computer Science*, 519:68–87. Algorithmic Learning Theory.

[do Prado Leite and Doorn 2004] do Prado Leite, J. C. S. and Doorn, J. H. (2004). *Perspectives on Software Requirements: An Introduction*, pages 1–5. Springer US, Boston, MA.

[Feldt et al. 2018] Feldt, R., de Oliveira Neto, F. G., and Torkar, R. (2018). Ways of applying artificial intelligence in software engineering. In *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pages 35–41.

[Gramajo et al. 2020a] Gramajo, M., Ballejos, L., and Ale, M. (2020a). Seizing requirements engineering issues through supervised learning techniques. *IEEE Latin America Transactions*, 18:1164–1184.

[Gramajo et al. 2020b] Gramajo, M. G., Ballejos, L. C., and Ale, M. (2020b). Hacia la evaluación automática de la calidad de los requerimientos de software usando redes neuronales long short term memory. In *Workshop on Requirements Engineering (WER)*.

[ISO/IEC 2018] ISO/IEC (2018). *ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering*. ISO/IEC.

[Lima et al. 2019] Lima, M., Valle, V., Costa, E., Lira, F., and Gadelha, B. (2019). Software engineering repositories: Expanding the promise database. In *SBES 2019: Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 427–436.

[Pohl 2010] Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, Incorporated, 1st edition.

[Rich 1985] Rich, E. (1985). Artificial intelligence and the humanities. *Computers and the Humanities*, 19(2):117–122.

[Samuel 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

[Souza et al. 2020] Souza, J. H. J., Marques, L. C., Conte, T. U., and Zaina, L. A. M. (2020). Descrevendo requisitos de user experience em critérios de aceitação de user stories. In *Workshop on Requirements Engineering (WER)*.

[Thayer et al. 1997] Thayer, R. H., Bailin, S. C., and Dorfman, M. (1997). *Software Requirements Engineerings, 2nd Edition*. IEEE Computer Society Press, Washington, DC, USA, 2nd edition.

[Wiesweg et al. 2020] Wiesweg, F., Vogelsang, A., and Mendez, D. (2020). Data-driven risk management for requirements engineering: An automated approach based on bayesian networks.

[Yang et al. 2012] Yang, H., De Roeck, A., Gervasi, V., Willis, A., and Nuseibeh, B. (2012). Speculative requirements: Automatic detection of uncertainty in natural language requirements. In *2012 20th IEEE International Requirements Engineering Conference (RE)*, pages 11–20.