Text Classification for Problem Diagnosis Based on Prior Service Calls Descriptions: A preprocessing comparison approach

Daniel Ochoa, Thiago Yoshimura

Key words:

Abstract:

In the realm of Natural Language Processing (NLP), effective text classification plays a pivotal role in organizing vast amounts of textual data. This paper focuses on the challenges inherent in classifying elevator-related issues within a Colombian elevator maintenance company, navigating issues like limited domain documentation, error-prone manual reports, and the intricacies of processing Spanish text. Our approach involves adapting NLP preprocessing techniques and innovative strategies to enhance classification performance.

Results showcase the delicate balance between performance, time, and feature reduction. Stemming emerges as a preferred technique, offering improved accuracy with a manageable increase in processing time. The integration of Latent Semantic Analysis (LSA) for feature reduction highlights its potential benefits, albeit with a trade-off in classification accuracy. Data translation experiments reveal that translating from Spanish to English minimally impacts accuracy but significantly reduces features due to grammatical error corrections.

This study contributes insights into effective text classification in complex domains, emphasizing the importance of balancing preprocessing techniques and considering trade-offs. Stemming proves advantageous, while LSA requires careful consideration. Data translation insights provide practical considerations for language translation techniques in NLP workflows. Overall, this research contributes to the advancement of text classification methodologies, setting the stage for improved information organization in real-world applications.

1. Introduction:

In an age saturated with information, effectively organizing and categorizing vast amounts of textual data has become crucial. This paper explores text classification, a key part of Natural Language Processing (NLP) that involves assigning predefined categories or labels to unstructured text. The focus of our study lies in addressing the intricate challenges associated with the classification of elevator-related issues within a Colombian company specializing in elevator maintenance. These challenges include the absence of a well-documented domain, reliance on manually crafted reports prone to grammatical errors, and the inherent complexity of processing a language like Spanish, which lacks an abundance of examples compared to its English counterpart. To navigate these challenges, our approach involves the adaptation of fundamental NLP preprocessing concepts tailored specifically to this database. Therefore, we aim to contribute to this domain by analyzing the performance of combinations of different preprocessing techniques in respect to accuracy, time and quantity of features. Also, this experiment tries to enhance the model's performance by employing unconventional strategies such as text translation from Spanish to English and leveraging Latent Semantic Analysis (LSA) for feature reduction. Through these innovative techniques, we endeavor to contribute to the advancement of text classification methodologies in complex and underexplored domains. Lastly, we aim to contribute to the NLP text classification area, by summarizing the literature about this subject highlighting some ideas and publications that we understand that contributed to achieve our results.

2. Literature Review

2.1 Background Concepts

Currently, an overwhelming volume of textual data is being produced, spanning articles, reports, social media posts, news articles, and emails, as text continues to be the most conventional and uncomplicated means of information transmission. The most effective approach to automating text classification is by employing machine learning techniques (Agarwal 2018). Text classification, as Sebastiani (2002) describes it, involves the automated task of assigning predefined categories or labels to text documents based on their content. This entails training a machine learning model to recognize patterns and connections within the text data, allowing the model to categorize unseen documents effectively. Similarly, according to Kadhim (2019), text classification is a fundamental technique for efficient information organization. It accomplishes this by automatically slotting a particular document into predefined categories.

Text classification involves a multistep process, with preprocessing, feature extraction, and feature selection playing integral roles (Kadhim, A. I., 2019). Preprocessing in text classification refers to the crucial initial step of preparing and cleaning raw textual data before it is used to train a machine learning model. (Kannan S., et al, 2014). This essential process involves a series of operations such as tokenization, lowercasing, removing punctuation, stop words, and special characters, as well as stemming or lemmatization to standardize words. (B. Magnini, C. Strapparava, 2002).

Feature extraction is a crucial step in machine learning and data analysis. It helps convert messy text and document data into a structured format, using mathematical models. (Dalal, M. K., et al, 2011). Feature extraction plays a pivotal role in providing valuable information, such as identifying the predominant terms within each text. The process of selecting and representing these keywords holds a huge significance in the area of supervised machine learning. The choice of keywords significantly impacts the classification technique's capacity to discern and highlight the most crucial patterns. (Kadhim, A. I., 2019).

Feature selection in terms of text classification is a process that aims to identify a certain number of important text features to achieve minimum text classification error (Sebastiani 2002). Feature selection offers several benefits, including data size reduction, decreased storage requirements, enhanced prediction accuracy, prevention of overfitting, and faster execution and training times, all while making variable interpretation easier. (Rizgar R. et. al, 2020). Feature selection methods can be broadly categorized into two main groups: the filter model, wrapper model The wrapper model uses a predefined learning algorithm for feature selection and assesses feature performance to make informed selections (John et al., 1994). On the other hand, the filter model relies on intrinsic attributes of the training data to autonomously identify and select certain features, without relying on any particular learning algorithm (John et al., 1994). There is also a third technique called embedded model; embedded techniques share similarities with wrapper methods, with the key distinction that they incorporate feature selection as an integral part of the training process (D. Roobaert, et. al, 2008).

Lastly, the categorization technique is applied through a specific machine learning model. The categorization technique is an effective approach used to create patterns for sorting data into different income groups. This method relies on supervised machine learning to identify patterns that link keyword groups with income data categories (Kadhim, A. I., 2019). Different algorithms can be applied to solve the problem of text categorization, from traditional algorithms such as Logistic Regression, Naive Bayes or Random Forest, until more complex ones such as CNN and RNN.

2.2 Related Surveys

Toman, Tesar and Jezek (2006) developed a project focusing on the comparison of diverse lemmatization and stemming algorithms used in NLP and also presented a novel lemmatization algorithm. They created a lemmatization dictionary based on EWN thesaurus, a multilingual database of words and relations between them. They used it as an interconnection between two different languages (English and Czech), resulting in reference indexes. They used two datasets, one with 8000 documents in English and the second with 8000 documents in Czech, and 5 categories which are similar between the datasets. Lastly, the performance of this new lemmatization technique was compared with other techniques. The results showed that the proposed technique achieved slightly lower performance than others, however it reduced the number of

features by 50%. Also, the results exposed that, for this dataset, no data preprocessing was better than any preprocessing technique.

Saad and Ashour (2010) proposed a new combination to be applied in the Arabic text using a dataset with 119 text documents connected with tree categories (Health, Sport, and Computer & communications) by using different techniques of preprocessing like TF Transform, IDF Transform, TF IDF Transformation, minTermFreq, normalizeDocLength, outputWordCounts, Stemmer and using a decision tree as a method of classification. They found that the results could vary from one dataset to another, however, the use of wc-norm and wc- tfidf brought a good performance and accuracy. In addition, they found a dramatically reduced dimensionality combined with the stemming and pruning enhanced the accuracy and performance of their classification.

Uysal and Gunal (2014) proposed an investigation on the impact of preprocessing on text classification by the multiple combinations of the methods: tokenization, stopword removal, lowercase conversion, and stemming, helped by using four datasets, two of them on emails and another two on news in both languages English and Turkish. They decided to apply a support vector machine(SVM) and make a variation of the feature size including 10, 20, 50, 100, 200, 500, 1000, and 2000 word unigrams. They concluded that depending on the domain and language used in the datasets appropriate combinations of preprocessing methods should be selected because they provide an improvement in classification accuracy. In contrast, other varieties could decrease the accuracy. For their specific case, they found that lowercase conversion regardless of the language could improve classifications in terms of accuracy. Furthermore, they found that using stop words is an important task that in previous studies was taken as irrelevant.

Pradana and Hayaty (2019) proposed four preprocessing conditions by the combination of the stopword and the steaming methods with a focus on the Indonesian language by getting data from Twitter and applying an SVM model to measure and compare the accuracy scores. In their experiments, they found that the score in accuracy for the data that was passed through the stemming got a higher performance compared to the one that passed through the stopword. However, the difference between them did not allow them to conclude what method was better and proposed future investigations to attempt to add lemmatization as a method to be compared.

Kerner, Miller and Yigal (2020) studied the impact of all possible combinations for basic preprocessing methods for text classification by selecting six (spelling correction, H–HTML tag removal, converting uppercase letters into lowercase letters, punctuation mark removal, reduction of repeated characters, and stopword removal)and applying them to four benchmark texts (WebKB, R8, SMS Spam Collection, Sentiment Labelled Sentences), additionally naive Bayes, support vector machines, and random forest were selected as the three machine learning algorithms to compare the results. According to their experiment, they found that in four of the datasets, the stopword did not give any beneficial result, however, the conversion to lowercase preprocessing on its own shows

an improvement. In general, they proved that at least one combination could improve the general accuracy for each domain.

2.3 Literature Review Conclusion

In the field of elevator maintenance, corrective maintenance plays a critical role, addressing customer needs when their equipment encounters issues. The technical personnel, upon making the necessary corrections, generate a detailed report for the maintenance company, which becomes an essential tool for planning future preventive maintenance. Since the information is conveyed through a technical message, it is imperative that it be classified clearly and precisely to facilitate a general understanding throughout the department. This classification not only optimizes internal communication but also enables more informed and strategic decision-making at the company level, ensuring efficient resource management and continuous improvement in maintenance operations.

The central objective of this research is to develop a robust text classification algorithm designed to categorize messages authored by technicians within specific problem domains. This project aims to work with authentic data collected from an elevator maintenance company based in Colombia. The real-world nature of this data presents unique challenges, as it is often messier and less structured than benchmark databases typically used in similar research endeavors. While the literature review provided a foundation for understanding text preprocessing concepts, it is noteworthy that no prior research has been identified in the domain of text classification specifically tailored to elevator maintenance. In contrast, there is a body of research focused on classification within maintenance contexts in other industries.

In addition to the challenge of dealing with actual maintenance data, this project is further complicated by its reliance on a Spanish-language database. While Spanish is widely spoken, its specialized or less common vocabulary may have limited training data available, potentially affecting the accuracy of lemmatization for such words. Furthermore, Spanish presents morphological complexity due to its highly inflected nature, featuring an intricate system of verb conjugation and noun declension. Spanish verbs, for instance, can manifest numerous forms contingent upon tense, mood, person, and number, making lemmatization a more intricate endeavor.

This research project attempts to develop innovative solutions to address these challenges and contribute valuable insights to the field of elevator maintenance, ultimately enhancing maintenance efficiency and resource utilization.

2.4 Project Context

Maintenance can generally be categorized into two primary types: corrective maintenance and preventive maintenance. While corrective maintenance often provides limited insights for decision-making, companies tend to prioritize preventive measures to reduce the need for corrective actions.

In the realm of elevator maintenance, a significant number of issues arise that are corrective in nature. Interestingly, these issues may not always be attributed to the elevator's machinery or the quality of preventive maintenance services, it could also be a user issue. Within corrective maintenance, it is essential to distinguish between problems external to the elevator and those directly related to it. For a comprehensive understanding, it is imperative to search deeper and categorize these issues, enabling companies to make more informed decisions.

Gathering this data involves a multi-step process. Initially, the technician who attends to the service drafts a report detailing the identified issues and their respective solutions. This report is then vetted by a supervisor for accuracy and feasibility. Subsequently, the quality department classifies each report into one of several categories. This classification is typically performed manually, requiring individuals to interpret the technician's descriptions, understand their unique expressions, and navigate potential grammatical errors. This process can be both labor-intensive and time-consuming.

Once the information is systematically categorized, analyzing the patterns and trends of corrective maintenance becomes more straightforward. Such analysis aids in making precise decisions to preempt future complications. Thus, accessing this data in real-time could facilitate adaptive strategies based on emerging results. To streamline this process, we propose a system for classifying reports. This system would account for technicians' varied expressions and potential grammatical inaccuracies, emphasizing the importance of effective preprocessing.

3. Methodology

In this session, the pipeline of the model is proposed. However, before starting, the data pass through a translation phase, because this research tries to understand the differences in results testing the original data in Spanish and the translated data in English. The translation is made by a package in python called Googletrans.

After this part, both databases follow the same pipeline. The first part is called the preprocessing part, where the following steps are made - tokenization, stopwords removal, lowercase conversion, lemmatization, and stemming.

The tokenization step is made using NLTK's word_tokenize function. Tokenization is the process of breaking down a text into smaller units, which are typically words or subwords. The word_tokenize('punkt') is a function provided by NLTK that takes a text string as input and returns a list of tokens (words). The 'punkt' module contains various pre-trained tokenizers, including a word tokenizer.

The next step is the stopwords removal. NLTK (Natural Language Toolkit) provides a convenient way to download a list of stopwords and remove them from your text data. NLTK includes a list of stopwords for various languages. Therefore for this part the functions used are stopwords.words('english') and stopwords.words('spanish').

The following step is the lowercase conversion. In this experiment, the lower() method is used to convert the entire text to lowercase. This operation is useful in text preprocessing to ensure that the text is consistent and doesn't treat "example" and "EXAMPLE" as different words, for example.

Stemming and lemmatization are both text normalization techniques used in natural language processing (NLP) to reduce words to their base or root forms. While they serve similar purposes, they are distinct processes, and they are not typically used together in the same NLP pipeline. Therefore, this paper explores the uses of stemming and lemmatization separately in the pipeline, but also togheter.

The following step is the lemmatization. Lemmatization is the process of reducing words to their base or dictionary form, called "lemmas." NLTK provides tools to perform lemmatization on text data. Before using lemmatization, it is necessary to download the WordNet dataset, which is a lexical database used by NLTK for lemmatization. The lemmatizer.lemmatize(word) method takes each word and returns its lemma. The output will be the base or dictionary form of each word, which is often more meaningful for text analysis. To perform lemmatization in Spanish using NLTK, you can use the Pattern library along with the NLTK interface. NLTK's WordNet is not used for Spanish lemmatization; instead, we rely on the Pattern library for this purpose.

To perform stemming using NLTK, it is possible to use various stemmers available. Stemming is the process of reducing words to their root or base form. The stemming method can be used on top of the previous languages datasets, such as the 'punkt' dataset (for tokenization) and the WordNet corpus (for the WordNet Lemmatizer). This experiment uses the Snowball Stemmer, also known as the Porter2 Stemmer, that is a popular stemmer in NLTK. Snowball Stemmer can be used for other languages also, therefore this research sets the parameter to Spanish for the Spanish database.

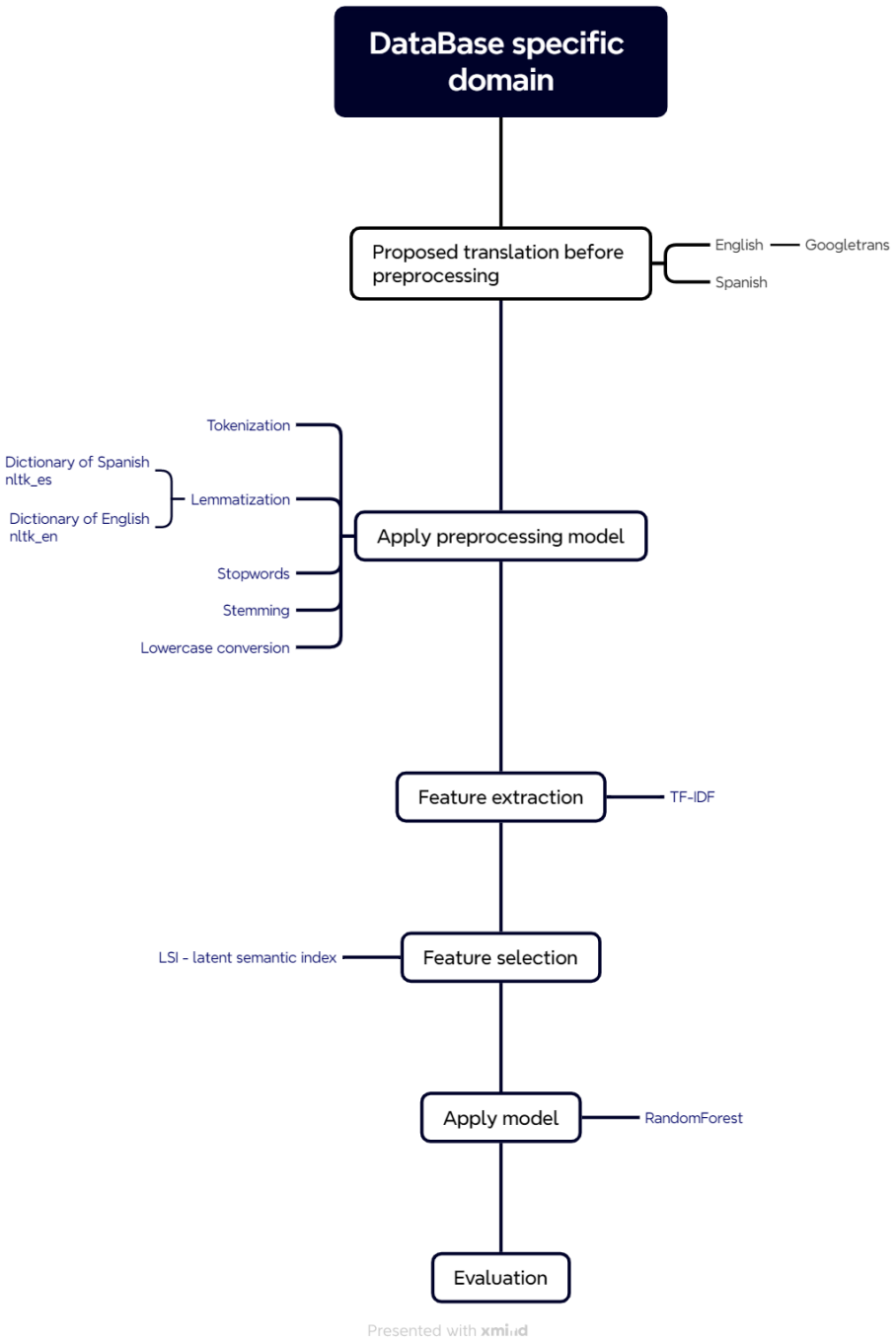
To perform feature extraction using TF-IDF (Term Frequency-Inverse Document Frequency) in Python, this experiment uses the popular library scikit-learn. Scikit-learn provides a TfidfVectorizer class for this purpose.

First it fits the TF-IDF vectorizer on the training data, which learns the vocabulary and computes the TF-IDF values for each term in the training documents. The result is the TF-IDF representation of the training data. Secondly, it transforms the test data using the TF-IDF vectorizer that has already been fitted on the training data. This step ensures that the same vocabulary and TF-IDF values are applied to both the training and test data, maintaining consistency. Finally, The X_train_tfidf and X_test_tfidf matrices contain the TF-IDF features for the training and test data, respectively. These matrices can be used as input to the machine learning model.

The next step is the feature selection session. Latent Semantic Analysis (LSA), also known as Latent Semantic Indexing (LSI), is a technique used for dimensionality reduction and extracting latent semantic information from textual data. It's often

employed in natural language processing (NLP) and information retrieval tasks. This experiment uses scikit-learn's TruncatedSVD class to perform LSA on the TF-IDF matrix.

To perform the text classification step, this experiment uses Random Forest approach.



5. Results

In this section, we present the results of our experimentation with various text processing approaches applied to a machine learning pipeline. The performance of each approach is evaluated using essential metrics such as F1 score and accuracy. The table below summarizes the outcomes, showcasing the impact of different text preprocessing techniques on the model's predictive capabilities.

We conducted experiments on a dataset using different combinations of text preprocessing techniques. The primary focus was on lowercasing, punctuation removal, stopword removal, lemmatization, and stemming. Additionally, we examined the effect of dimensionality reduction using Latent Semantic Indexing (LSI) with 70 components. The performance of each configuration was evaluated using F1 score, accuracy, time and number of features.

The following preprocessing techniques were used:

- TF-IDF Vectorization (tfidf):
  The baseline TF-IDF vectorization was employed as a reference. It provides a standard representation of text features.

- Lowercasing (lower) and Ponctuation Removal (ponct):
  Lowercasing and punctuation removal were applied to explore the impact of basic text normalization.

- Stopword Removal (stopw):
  The removal of stopwords aimed to assess the significance of common words in the dataset on classification performance.

- Lemmatization (lemma):
  Lemmatization was used to reduce words to their base or root form, potentially improving the model's ability to generalize.

- Stemming (stemm):
  Stemming was employed to further reduce words to their stems, simplifying the vocabulary and potentially aiding in capturing semantic similarities.

- LSI (lsi):
  Latent Semantic Indexing (LSI) was explored to understand the impact of dimensionality reduction on classification performance. 70 components were used to compare to the other methods. 70 was chosen because it performed better than other numbers.

For the details of the code developed, it is located in the GitHub repository:
https://github.com/Dopo0/Text-Classification

5.1 Results of preprocessing techniques

Table 1 - Results for combinations of preprocessing techniques

Index	Description	# of Features	Pipeline Time	F1 Score	Accuracy
1	tfidf	19,535	14.826	0.6904	0.7110
2	lower-tfidf	19,535	14.819	0.6873	0.7082
3	lower-ponct-tfidf	20,581	17.450	0.6903	0.7109
4	lower-ponct-stopw-tfidf	20,417	14.845	0.6983	0.7161
5	lower-ponct-stopw-lemma-tfidf	16,447	141.918	0.7110	0.7289
6	lower-ponct-stopw-stemm-tfidf	11,914	24.162	0.7198	0.7375
7	lower-ponct-stopw-lemma-stemm-tfidf	11,737	151.150	0.7193	0.7358

Quantity of Features
Approaches 5, 6, and 7 exhibit fewer features, suggesting effective dimensionality reduction through stemming and lemmatization. Comparing approaches 5 (lemmatization) and 6 (stemming) with 4 (no normalization), stemming generates more reductions (-45%) than lemmatization (-20%).

Removing punctuation showed an increase in features, contrary to expectations. Analysis revealed frequent grammatical errors involving unions of punctuation and words, causing words to be split and generating more features.

Pipeline Time
Pipeline time reflects the computational cost of each approach. Notably, approaches 6 and 7 have the longest pipeline times (141 and 151 seconds), primarily due to the inclusion of lemmatization. Conversely, approaches 2, 3, and 4 maintain relatively short pipeline times (from 14 to 17 seconds), indicating that lowercase, punctuation, and stop words removal do not represent a high computational cost. Additionally, approach 6 (stemming) did not result in a significantly higher time increase compared to lemmatization, supporting the preference for stemming over lemmatization.

F1 Score and Accuracy

While differences in F1 scores and accuracy are subtle, certain trends emerge. Approaches involving stemming tend to perform marginally better. The highest F1 score and accuracy are achieved by approaches 6 (F1: 0.7198, Accuracy: 0.7375) and 7 (F1: 0.7193, Accuracy: 0.7358), indicating the potential benefits of a holistic text processing strategy.

Performance vs. Time
The inclusion of lemmatization and stemming may enhance accuracy and F1 score, but it comes at the cost of increased processing time. While lemmatization results in a considerable processing time increase (+1200%), using only stemming represents a more manageable increase (+91%). We recommend using stemming due to its favorable balance between performance and time compared to not using the technique.

Performance vs. Feature Reduction
Feature reduction is somewhat correlated with performance, as fewer features due to lemmatization and stemming produce higher performance scores, indicating a reduction of noise in the data. The research proposes the use of latent semantic indexing (LSI) for dimension reduction to potentially further improve performance by reducing the number of dimensions.

5. 2 Results of preprocessing technique using LSI

Table 2 - Results for combinations of techniques adding LSI dimension reduction.

Description	# of features	Pipeline time	F1 score	Accuracy
lower-ponct-tfidf-lsi70	70	7.710	0.6048	0.6264
lower-ponct-stop-tfidf-lsi70	70	9.416	0.6268	0.6446
lower-ponct-stop-stemm-tfidf-lsi 70	70	18.783	0.6599	0.6752
lower-ponct-stop-lemma-stemm-tfidf-lsi70	70	129.203	0.6628	0.6784

LSI - Feature Reduction
The introduction of LSI results in a substantial reduction in the number of features, limiting the dimensionality of the dataset. This reduction is particularly notable, as seen in all four approaches where the number of features is fixed at 70.

The controlled number of features in LSI can be advantageous for handling high-dimensional data, potentially mitigating the curse of dimensionality and focusing on the most relevant information.

LSI - Pipeline Time Impact
LSI introduces an additional computational cost to the preprocessing pipeline, however it decreases the time that the model (random forest) takes to perform the classification. We can see that overall time is reduced compared to the counterparts with no LSI.

LSI - Performance
The integration of LSI is not without trade-offs, as evidenced by a discernible reduction in both F1 score and accuracy when compared to their non-LSI counterparts. The decrease in accuracy across equivalent approaches with and without LSI hovers in the range of 8% to 12%.
This reduction in performance metrics emphasizes the delicate balance that must be struck when implementing LSI. While feature reduction is achieved, there is a consequential compromise in classification accuracy.

5.3 Results for data translation

The last part involved training the model again using a translated version of the database. In this scenario, the accuracy was almost the same—less than 1% difference—between the Spanish dataset and its English-translated counterpart. However, the English dataset exhibited a significant drop in the number of features, ranging from 40% to 60% fewer features than the original dataset, depending on the combination of techniques used. We understand that many grammatical errors were corrected after translation, making the translation technique act as a feature reduction method.

Additionally, the technique of translating the database was very time-consuming because the translation process was performed row by row using a package that interacts with the Google Translator API. In our case, we do not recommend using this technique in the current operational workflow. However, in other scenarios, such as translating data in real-time after inserting it into a spreadsheet, this approach could be very useful.

6. Conclusion

In this study, we delved into the challenges of text classification within the specific domain of elevator-related issues in a Colombian company specializing in elevator maintenance. Our primary focus was on addressing the complexities associated with a lack of well-documented domain knowledge, grammatically error-prone manual reports, and the intricacies of processing Spanish text. The proposed methodology involved adapting Natural Language Processing (NLP) preprocessing techniques and exploring innovative strategies to enhance text classification performance.

The results of our experiments with various preprocessing techniques shed light on the trade-offs between performance, time, and quantity of features. Stemming, in particular, emerged as a favorable approach, striking a balance between improved accuracy and a manageable increase in processing time compared to lemmatization. The inclusion of Latent Semantic Analysis (LSA) for feature reduction demonstrated its potential to address the curse of dimensionality, although it came at the cost of a noticeable reduction in classification accuracy.

Furthermore, our exploration of data translation revealed that translating the database from Spanish to English had a minimal impact on accuracy but significantly reduced the number of features. This reduction was attributed to the correction of grammatical errors during the translation process, acting as an unintended feature reduction method. However, the time-consuming nature of the translation process raised practical considerations, suggesting limited applicability in certain operational workflows.

In conclusion, this study contributes insights into effective text classification strategies, especially in complex and underexplored domains. The findings emphasize the importance of carefully balancing preprocessing techniques, considering trade-offs between performance and computational costs. While stemming proves advantageous in this context, the integration of LSA requires careful consideration due to its impact on classification accuracy. Additionally, the exploration of data translation provides valuable insights into the potential benefits and challenges of leveraging language translation techniques in NLP workflows. This research sets the stage for further refinement and optimization of text classification methodologies in specialized domains, paving the way for improved information organization and categorization in real-world applications.

Reference:

Aggarwal, C. C. (2018). Machine Learning for Text. Springer Publishing Company,Incorporated, 1st edition.

Magnini, B., Strapparava, C., Pezzulo, G., & Gliozzo, A. (2002). The role of domain information in word sense disambiguation. Natural Language Engineering, 8(4), 359-373.

Dalal, M. K., & Zaveri, M. A. (2011). Automatic text classification: a technical review. International Journal of Computer Applications, 28(2), 37-40.

Roobaert, D., Karakoulas, G., & Chawla, N. V. (2006). Information gain, correlation and support vector machines. In Feature extraction: Foundations and applications (pp. 463-470). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., & Saeed, J. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. Journal of Applied Science and Technology Trends, 1(2), 56-70.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In Machine learning proceedings 1994 (pp. 121-129). Morgan Kaufmann.

Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. Artificial Intelligence Review, 52(1), 273-292.

Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S., & Gurusamy, V. (2014). Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks, 5(1), 7-16.

HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. PloS one, 15(5), e0232525.

Pradana, A. W., & Hayaty, M. (2019). The effect of stemming and removal of stopwords on the accuracy of sentiment analysis on indonesian-language texts. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 375-380.

Saad, M. K., & Ashour, W. (2010, November). Arabic text classification using decision trees. In Proceedings of the 12th international workshop on computer science and information technologies CSIT (Vol. 2, pp. 75-79).

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47.

Toman, M., Tesar, R., & Jezek, K. (2006). Influence of word normalization on text classification. Proceedings of InSciT, 4(354-358), 9.

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. Information processing & management, 50(1), 104-112.