ISSN: 2317-0840

Comparison between document vectorization methods: a case study for textual data

Jéssica Kubrusly[1†], Gabriel G. L. Valenotti[2],

[1] *Departamento de Estatística, Instituto de Matemática e Estatística, Universidade Federal Fluminense.*
[2] *Instituto de Computação, Universidade Federal Fluminense.*

Abstract: *The explosion of digital information in recent decades has brought a massive volume of text data. The interest in extracting knowledge from this vast amount of data gave rise to Text Mining. One of the challenges in this field is to transform a text corpus into a numerical database. This process, called document vectorization, is crucial for automating information extraction. The goal of this work is to compare the performance of four document vectorization methods when used for classification purposes. The compared vectorization methods were Bag of Words (BoW), TF-IDF, and two different architectures of doc2vec, CBOW and skip-gram. The classification methods applied were Logistic Regression, Decision Tree, Random Forest, XGBoost, and Perceptron. The dataset used was the publicly available Women's E-Commerce Clothing Reviews dataset, which consists of 10 attributes, with three of them considered in this work: the item review text, the review title, and a categorical variable indicating whether the customer recommends the product or not. A balanced random sample of 8,000 documents was selected, with 4,000 documents having positive recommendations and 4,000 with negative recommendations. This dataset was split into training (70%) and testing (30%) sets. The performance comparison metric was the area under the ROC curve (AUC). When comparing the document vectorization methods, both architectures of doc2vec outperformed the other vectorization methods across all tested classification methods.*

Keywords: *Text Mining; Doc2Vec; TF-IDF; Classificaton Methods.*

Comparação de métodos de vetorização de documentos: um estudo de caso com dados textuais

Resumo: *A explosão de informações digitais nas últimas décadas trouxe um enorme volume de dados em forma de texto. O interesse em extrair conhecimento desta vasta quantidade de dados originou a Mineração de Texto. Um dos desafios nesta área é transformar um banco de textos em uma base de dados numérica. Esse processo, chamado de vetorização de documentos, é fundamental para a automatização da extração de informação. O objetivo deste trabalho é comparar o desempenho de quatro métodos de vetorização de documentos quando utilizados para fins de classificação. Os métodos de vetorização comparados foram: BoW, TF-IDF e as duas arquiteturas diferentes do doc2vec, CBOW e skip. Os métodos de classificação aplicados foram: Regressão Logística, Árvore de Classificação, Floresta Aleatória, XGBoost e Perceptron. A base de dados foi a base pública The Women's E-Commerce Clothing Reviews, composta por 10 atributos, entre os quais 3 deles foram considerados neste trabalho: o texto de avaliação do item, o título da avaliação e uma variável categórica que indica se o cliente recomenda ou não o produto. Uma amostra aleatória balanceada de 8.000 documentos, 4.000 documentos com recomendação positiva e 4.000 com recomendação negativa, foi sorteada e dividida em treino (70%) e teste (30%). A medida de comparação de desempenho foi a área embaixo da curva ROC (AUC). Quando comparados os métodos de vetorização de documentos, as duas arquiteturas do doc2vec apresentaram resultados superiores às demais em todos os métodos de classificaçãp testados.*

Palavras-chave: *Mineração de Texto; Doc2Vec; TF-IDF; Métodos de Classificação.*

[†]Autor correspondente: ⟨jessicakubrusly@id.uff.br⟩.

Introduction

It is true that most Internet data is in unstructured form, primarily text. These originate from social networks, such as Facebook and Twitter, or even corporate data, such as complaints and opinion polls. A real example is The Women's Clothing E-Commerce Reviews1, which consists of reviews written by real customers. Text Mining, a Data Mining branch, has arisen from the need to process and extract information from this large mass of textual data.

A Sentiment Analysis comprises the computational treatment of opinions, sentiments and text subjectivity (MEDHAT; HASSAN; KORASHY, 2014). Its techniques can be roughly categorized into two groups, the first consisting in the Lexicon-based Approach, which associates words or expressions with positive, negative or neutral feelings, and the second comprises the Machine Learning Approach.

Concerning the Machine Learning Approach, documents must be vectorially represented so that they can be used as input data for Machine Learning methods. The most intuitive alternative is to work with term (word) presence or frequency in each document (BREIMAN, 2001). However, there are other ways of document vectorization that will be addressed in this work.

In this context, this study analyzed The Women's Clothing E-Commerce Re-views database, which consists of consumer comments regarding a particular clothing item and a label designating whether or not the consumer indicates said item. Aiming to automatically predict customer recommendations based on their textual reports, the goal of this work was to compare the quality of different document vectorization methods used as input data for classification methods.

The paper is organized as follows. Section 2 cites some related studies, Section 3 describes the main Text Mining process steps and Section 4 presents the applied classification methods, namely Logistic Regression, Classification Tree, Random Forest, XGBoost and Perpeptron. Section 5 comprises quality measures for the classification methods, Section 6 described the database, Section 7 presents some interesting numerical results and analyses and, finally, the conclusions are reported in Section 8.

Related Work

In 2022, Kubrusly et al. (KUBRUSLY; NEVES; MARQUES, 2022) worked with the same dataset and used only the BoW vectorization method. The main objective of that study was to compare the performance of different tree-based classification methods when applied to classify the texts based on customer recommendations. The results indicate that Random Forest and XGBoost exhibit overfitting, the Classification Tree is proficient at detecting negative reviews but struggles with positive ones, and Gradient Boosting shows stable values with F1 measures above 77% for the test dataset. In contrast, the focus of this article is on comparing the vectorization methods, not the classification methods.

In 2008, Schütze, Manning e Raghavan (2008) discusse the TF-IDF and BoW models in detail, providing a solid foundation for traditional document vectorization techniques. These methods remain relevant and are often used as baselines in comparative studies. Five years later, Mikolov et al. (2013a) introduces Word2Vec, a method that generates word embeddings, vector representations of words. It has paved the way for numerous subsequent studies on word embeddings and their applications in text analysis. In 2014, Le e Mikolov (2014) propose an unsupervised algorithm that learns vector representations of sentences and text documents and Kim (2014) demonstrates the application of Convolutional Neural Networks (CNNs) to sentence and document classification tasks. It showcases the potential of deep learning models in document vectorization.

Qasem e Sajid (2022) performed research on fake news detection tools. They investigated and compared Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-

IDF) approaches, with N-grams, and three conventional machine classifiers, Support Vector Machine (SVM), Naive Bayes (NB), and Decision Tree (DT). It was demonstrated that the traditional models are still good candidates and that the use of bigram combined with BOW and DT classifier performs the best among others, with an accuracy of 99.74%.

Ling e Chen (2023) conducted a study employing tweets coming from a human and a bot. The interest was in comparing the performances of different word embedding methods and classification models. They used f1-score and confusion matrix for evaluation. The results suggest the power of Transformer based vectorization methods including Doc2Vec, BERT, and fastText when handling imbalanced data.

In the study of Joseph e Yerima (2022), a comparative analysis of popular word embedding techniques for SMS spam detection is provided by evaluating their performance on a publicly available ham and spam data set. The performance of the word embedding techniques is investigated using 5 different machine learning classifiers, i.e., Multinomial Naive Bayes (MNB), KNN, SVM, Random Forest, and Extra Trees. Based on the dataset employed in the study, N-gram, BOW, and TF-IDF with oversampling achieved the highest F1 scores of 0.99 for ham and 0.94 for spam.

Text Mining

Text Mining is the process that basically consists in the extraction of non-trivial patterns or knowledge from unstructured text documents. This process can be categorized into two main steps: refinement, which transforms the original textual database into a numerical database; and the information extraction process, which consists of detecting patterns from the refined database using conventional statistical tools (FRITSCH; GUENTHER; WRIGHT, 2019).

Text Preprocessing

The main refinement steps of a textual database, also called preprocessing techniques, are: Tokenization; Stop Word Removal; Normalization. A brief explanation of each is presented below.

Tokenization is the first preprocessing stage and aims to extract minimum text units from a free text. These units are called tokens and most often refer to a single word.

Stop Words are the most frequent terms in a language. They have no semantic value and only aid in the general understanding of the text. Stop words are usually characterized by articles, prepositions, punctuation, conjunctions and pronouns. A pre-established list is usually applied, called a stoplist. The removal of stop words considerably reduces the amount of tokens and improves the analysis to be performed.

Normalization is the process of grouping words that share the same pattern. The main normalization methods are stemming and lemmatization, and further explanations on these terms can be found in Goodfellow, Bengio e Courville (2016). The lemmatization method will be applied here in, which, for example, replaces the tokens “calculate”, “calculating” and “calculated” for the term “calculate”. This process is illustrated in Figure 1.

Bag of Words

Following these steps, each document was then transformed into a bag of terms (Bag of Words - BoW). Considering a textual database formed by n documents that together contain m terms. To reduce the dimension of this representation, term selection is performed. Term Selection, proposed by Hastie et al. (2009), establishes a set of significant database terms. Non-significant terms, which have low semantic value, appear at very high or very low frequencies in document sets and are not considered in the analyses.

Figure 1: A text preprocessing example.

Source: Authors.

After terms selection, considering a textual database formed by n documents that together contain p terms. The $n \times p$ matrix A, where each element $a_{i,j}$ represents the frequency with which the term j occurs in document i, is called the Document-term Matrix. Each line of this matrix is a vector representation of a document. Each column corresponds to a term and can be understood as a document attribute. So, the vector representation of documents in this methodology will be given by:

$$d_i^1 = (a_{i,1}, a_{i,2}, \ldots, a_{i,p}) \tag{1}$$

Figure 2: A bag of words vector representation example.

	love	dress	way	style	disappoint	quality	color	poor	soft	...
1	4	1	1	1	0	0	0	0	0	
2	1	1	0	1	1	2	1	1	0	
3	0	0	0	1	0	0	1	0	1	
4	1	2	0	0	0	0	0	0	0	
⋮										

nxp

Source: Authors.

TF-IDF

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a widely used technique in natural language processing and information retrieval. It serves as a powerful tool for quantifying the importance of terms (words or phrases) within a collection of documents. TF-IDF is particularly valuable for text analysis tasks such as document retrieval, text classification, and information retrieval.

TF (Term Frequency) measures the frequency of a term within a document. $TF_{i,j}$ represents how often the term j appears in the document i relative to the total number of terms in the

document i. Essentially, it quantifies the relevance of a term to a specific document.

$$\text{TF}i,j = \frac{\text{number of times } j \text{ appears in } i}{\text{number of terms in } i} = \frac{a_{i,j}}{\sum_{j=1}^{p} a_{i,j}}$$

IDF (Inverse Document Frequency), on the other hand, assesses the importance of a term across a collection of documents. IDF_j calculates the logarithmically scaled inverse of the fraction of documents containing the term j. Terms that are common across many documents receive a lower IDF score, while terms that are rare or unique receive a higher score.

$$\text{IDF}_j = \ln\left(\frac{\text{total number of documents}}{\text{number of documents with term } j}\right) = \ln\left(\frac{n}{\sum_{i=1}^{n} I_{\text{N}^\star}(a_{i,j})}\right)$$

where $\text{N}^\star$ is the set of positive integers and I is the indicator function.

By combining the TF and IDF components, the TF-IDF score for a term in a document is computed. This score highlights terms that are both frequent within the document and distinctive across the entire document collection. In practice, TF-IDF helps in identifying keywords or relevant terms that distinguish documents from one another, making it an essential technique for various text-based applications.

$$\text{TF-IDF}_{i,j} = \frac{TF_{i,j}}{IDF_j}$$

Thus, this methodology will provide the following vector representation of documents.

$$d_i^2 = (\text{TF-IDF}_{i,1}, \text{TF-IDF}_{i,2}, \ldots, \text{TF-IDF}_{i,p}) \tag{2}$$

Doc2Vec

Doc2Vec is a popular technique for document vectorization in natural language processing (NLP). It is an extension of Word2Vec and is designed to capture the semantic meaning of entire documents, making it a valuable tool for various NLP tasks. Unlike the previous methodologies, doc2vec uses as input the set of all m terms. In this methodology, no term selection is performed to reduce the base from m to p terms, since the dimensionality reduction is performed by the number of neurons in the hidden layer of the neural network.

Word2Vec, developed by Tomas Mikolov and his team (MIKOLOV et al., 2013a), revolutionized how computers understand and represent words in large text corpora. It operates on the principle that words with similar meanings often appear in similar contexts. The two architectures used in this work are Continuous Bag of Words (CBOW) and Skip-gram (MIKOLOV et al., 2013b).

The CBOW architectures predicts a target word based on its context words, it learns to understand a word by considering the words around it, Figure 3.

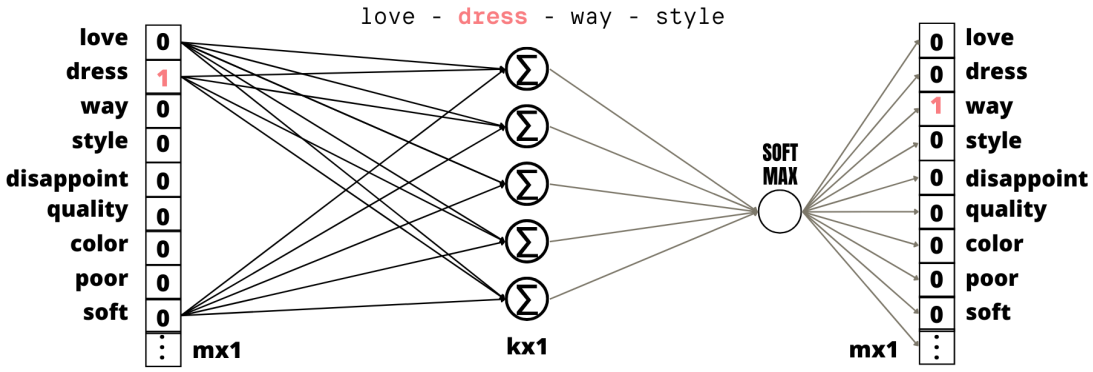
On the other hande, the Skip-gram architectures predicts context words based, it captures the context in which a word is used, Figure 4. In either of the two architectures, each word is represented by the synaptic weight values associated with the respective word.

After each word already has its vector representation provided by word2vec, represented by red lines in Figure 5, it is now possible to train the doc2vec network to establish a vector representation for the documents. Doc2Vec assigns a unique vector representation to each document in a corpus, it combines word vectors with document vectors during training, ensuring that the context of words within documents is preserved (Figure 5).

The documents vector representation will also be given by synaptic weights, but in this case, those associated with the documents

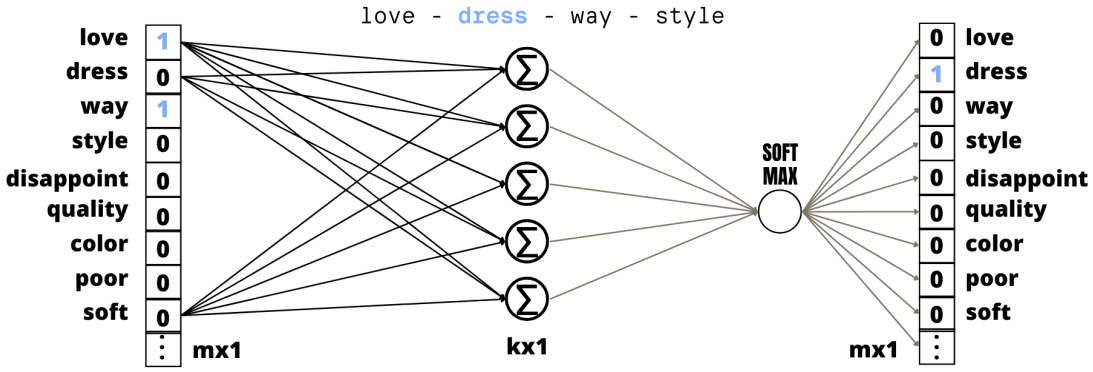
$$d_i^3 = \left(w_{i,1}^c, w_{i,2}^c, \ldots, w_{i,k}^c\right) \quad \text{and} \quad d_i^4 = \left(w_{i,1}^s, w_{i,2}^s, \ldots, w_{i,k}^s\right)$$

Figure 3: The CBOW architectures example.



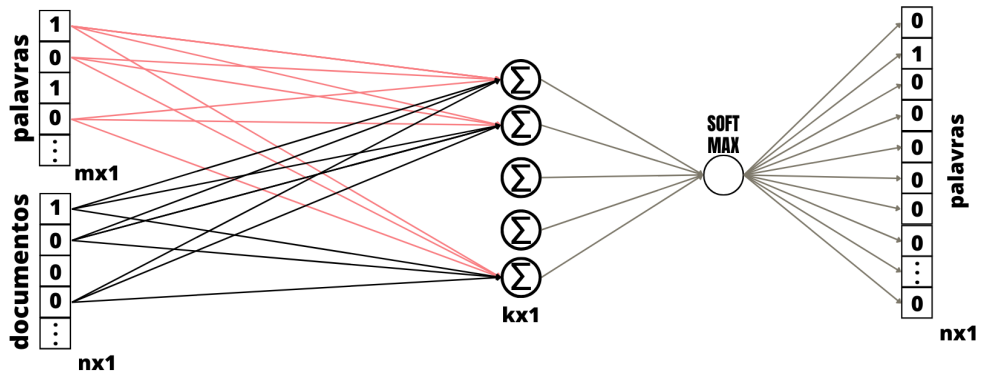
Source: Authors.

Figure 4: The Skip-gram architectures example.



Source: Authors.

Figure 5: The doc2vec architectures example.



Source: Authors.

where k is the number of neurons in the unique hidden layer, $w_{i,l}^{c}$ and $w_{i,l}^{s}$ are the synaptic weights from document i to neuron l considering the CBOW and Skip architectures, respectively.

To compare more fairly, the values of p, the number of terms after term selection, and k, the number of neurons in the hidden layer of the doc2vec neural network, are equal. Thus, all vector representations will have the same dimension, which will be referred to as p from now on.

Classification Methods

The main point of this work is to compare different document vectorization methods. To carry out this comparison, five different classification methods were used, which will be briefly explained in this section.

Considering a universe composed of n objects (documents) which are described by p attributes and each object i belongs to a known class $Y_i = \{0, 1\}$. A classification method aims to define a mathematical model capable of predicting the class of a new object when its p attributes are known. In practice, classification methods return a probability of belonging to the classes 1.

The Logistic Regression (MCCULLAGH, 2019; HASTIE et al., 2009) is a generalized linear model and a fundamental statistical and machine learning technique used for modeling binary outcomes or categorical data. Unlike linear regression, which predicts continuous numeric values, logistic regression is used when the dependent variable is binary or categorical. It models the probability of an event occurring, such as whether a customer recommend or not a product.

Introduced by Breiman et al. (1984), tree model is a classification method that uses the tree structure to recursively partition the data set. Once the input data has been split, the prediction is made from a simple classification method in each partition, such as the dominant class or the prevalence rate of the reference class.

The Random Forest (BREIMAN, 2001) is a bagging classification model created in order to improve the prediction of classification tree models. It consists of a collection of classification trees where each tree is constructed from a smaller data set composed of $\tilde{n} < n$ objects. The $\tilde{n}$ objects are selected from a Bagging strategy, such as Bootstrap Sampling (SUTTON, 2005). For each tree in the forest, at each split, a random selection of $\tilde{p} < p$ from the p variables is made. Only the $\tilde{p}$ selected variables will be considered in this partition. After many trees are generated, these results are combined to provide a final prediction.

The Random Forest method is a bagging algorithm. In these algorithms, trees are grown in parallel to obtain the average prediction across all trees. Gradient boosting, on the other hand, employs a sequential approach in obtaining predictions. In the Gradient Boosting method, each decision tree predicts the error of the previous one (AYYADEVARA, 2018). The XGBoost method is an upgraded Gradient Boosting Tree algorithm that can flexibly process sparse data and missing values (LIN, 2020). The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. It also incorporates a regularized model to prevent overfitting (CHEN; GUESTRIN, 2016).

The Support Vector Machine (SVM) method was developed by Cortes e Vapnik (1995). It is a powerful machine learning algorithm used for classification tasks. SVM works by finding a hyperplane that effectively separates data points belonging to different classes in a high-dimensional feature space using kernel functions. SVM can handle both linear and nonlinear data, and it is robust against overfitting when proper regularization is applied.

Finelly, the Multilayer Perceptron, also known as the Multilayer Neural Network (MLP) (GOODFELLOW; BENGIO; COURVILLE, 2016), is an extension of the simple Perceptron consisting of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to all neurons in the next layer, making it a deep learning model. This methods can capture complex relationships in the input data, making it suitable for more challenging classification and regression tasks. It's worth noting that the neural network model is exactly a linear logistic regression model in the hidden units, and all the parameters are estimated by maximum likelihood (HASTIE et al., 2009).

Quality Measures

The Receiver Operating Characteristic (ROC) curve is a fundamental quality measure used to assess the performance of classification methods. It provides valuable insights into the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) across different threshold settings. The ROC curve is particularly useful when evaluating binary classification models, such as those used in this study.

The ROC curve is created by plotting the true positive rate (TPR) on the y-axis against the false positive rate (FPR) on the x-axis at various threshold values. TPR represents the proportion of true positives correctly identified by the model, while FPR represents the proportion of false positives incorrectly classified as positive.

A perfect classification model would have an ROC curve that reaches the upper-left corner of the graph, indicating high sensitivity and low false positive rate, resulting in an area under the curve (AUC) equal to 1.0. Conversely, a random guessing model would have an ROC curve that closely resembles a diagonal line, with an AUC of 0.5.

The AUC, which stands for the Area Under the ROC Curve, is another crucial quality measure derived from the ROC curve. It quantifies the overall performance of a classification model. An AUC value of 1.0 indicates a perfect classifier, while an AUC of 0.5 suggests a model that performs no better than random chance.

An important point about ROC graphs is that they measure the ability of a classifier to produce good relative instance scores. A classifier need not produce accurate, calibrated probability estimates; it need only produce relative accurate scores that serve to discriminate positive and negative instances (FAWCETT, 2006).

The Dataset

The Women's Clothing E-Commerce Reviews was used as the dataset for this study and revolves around reviews written by customers. This dataset includes 23,486 rows and 10 feature variables. Each row corresponds to a customer review, and includes the following variables: Clothing ID; Age; Title; Review Text; Rating; Recommended IND; Positive Feedback Count; Division Name; Department Name; and Class Name. Of the 23,486 rows in the database, 19,314 refer to recommended items while the other 4,172 refer to non-recommended items.

Only three variables were considered herein: Review Text, string variable for the review body; Title, string variable for the title of the review; and Recommended IND, binary variable stating whether the customer recommends the product, where 1 is recommended and 0 is not recommended. Variables Title and Review Text were concatenated in order to add more wealth of information to the analysis. Then, only the text was used as a classification method attributes.

First, the data set was randomly split into 70% as a training set and 30% as a testing set. Since the original database contains many more recommended objects compared to non-recommended, not all of the documents should be used. To select balanced sets and respect the 70/30 ratio, the number of documents in the training set was 5,674 and the number of documents in testing was 2,445.

Results

This study was performed using the R Program (R Core Team, 2019). The `tidytext` (SILGE; ROBINSON, 2016), `tm` (FEINERER; HORNIK; MEYER, 2008) and `textstem` (RINKER, 2018) packages were used for textual preprocessing, BoW and TF-IDF representation. The `word2vec` (WIJFFELS, 2021b) and `doc2vec` (WIJFFELS, 2021a) packages were used for word and document embedding vector representations. The `rpart` (THERNEAU; ATKINSON, 2018), `randomForest` (LIAW; WIENER, 2002), `xgboost` (CHEN et al., 2023) and `neuralnet`

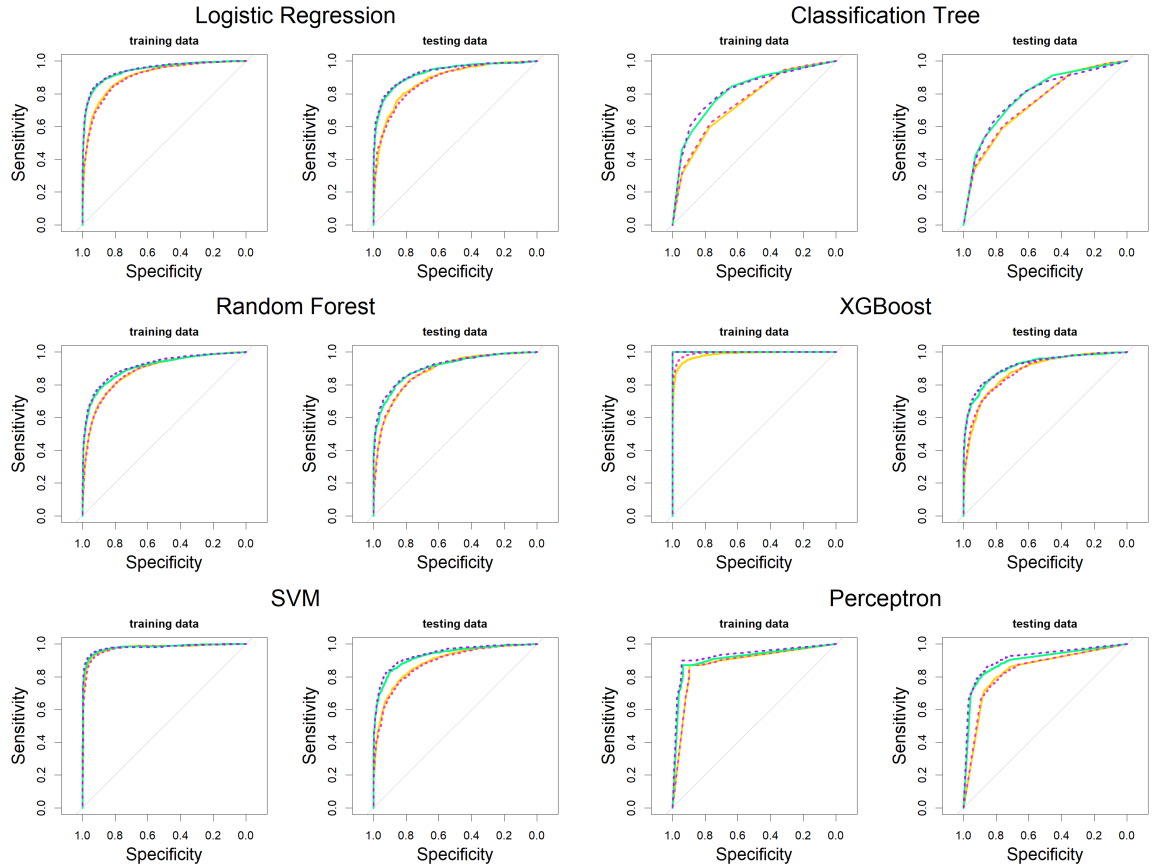
(FRITSCH; GUENTHER; WRIGHT, 2019) packages were used for the Classification Tree, Random Forest, XGBoost and Perseptron analyses, respectively. The `pROC` (ROBIN et al., 2011) package was used for ROC curve and AUC calculation.

The preprocessing described in Text Preprocessing Section was performed for the training data set. Stop words were removed and the normalization process was conducted based on Mechura's English lemmatization list. Following this process, the training textual database contained 5,675 documents and 7,066 different terms.

For the bag-of-words (BoW) and TF-IDF vector representations, a term selection process was performed to choose the top 200 most frequent terms. For both architectures of Doc2Vec, CBOW and Skip-gram, 200 neurons were used in the hidden layer. The result was that all vector representations considered in this case study had a dimension of 200.

All classification methods were run in R with their default parameter values. The multilayer perceptron model applied considered a single hidden layer with a single neuron. The SVM method was run with the radial kernel. For each one of the four vector representations, a total of 6 classification methods were run: Logistic Regression, Classification Tree, Random Forest, XGBoost, SVM and Perceptron.

Figure 6: ROC Curves for Classifications Methods



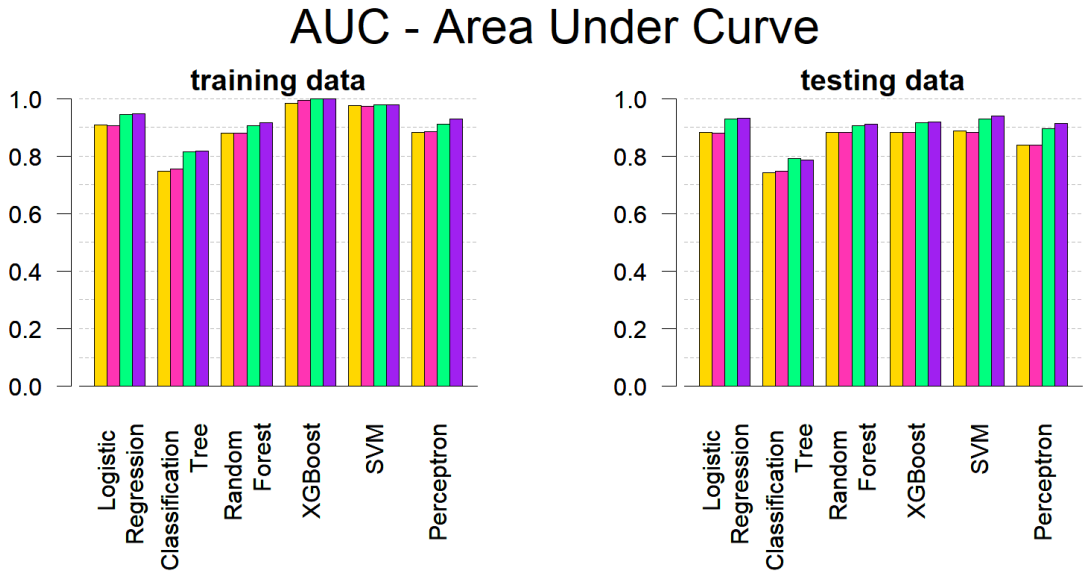
Source: Authors.

Figure 6 presents the ROC curves for all four vectorization methods and for each of the six classification methods. The BoW is indicates in yellow solid line, the TF-IDF in pink dashed line, Doc2Vec-CBOW in green solid line and Doc2Vec-Skip-gram in purple dashed line.

For all the classification methods tested in this study, the two architectures of doc2vec yielded better results when compared to BoW and TF-IDF. The ROC curves also indicate a similar performance of the classification methods when the input vectors were defined by BoW and TF-IDF. The performance of the classification methods for the two doc2vec architectures

was also similar.

Figure 7: AUC value for Classification Methods



Source: Authors.

Figures 7 presents barplots for the AUC value for each classification method and each vectorization method, both for the training and testing datasets. These figures confirm the previously presented result that the two Doc2Vec architectures yield and better performance. Furthermore, the vectorizations provided by BoW and TF-IDF yield similar results.

Conlusion

A text mining analysis via consumer reviews, i.e., free text referring to recommendable or not recommendable products, was performed. The goal was to compare the quality of classification methods when the input data was generated by different document vectorization methods. The vectorization methods compared in this work were: Bag of Words (BoW), TF-IDF, and two different architectures of Doc2Vec, CBOW and Skip-gram.

The classification methods applied were Logistic Regression, Decision Tree, Random Forest, XGBoost, SVM and Perceptron. The dataset used was the publicly available Women's E-Commerce Clothing Reviews dataset, which consists of 10 attributes, with three of them considered in this work: the item review text, the review title, and a categorical variable indicating whether the customer recommends the product or not. A balanced random sample of 8,000 documents was selected, with 4,000 documents having positive recommendations and 4,000 with negative recommendations. This dataset was split into training (70%) and testing (30%) sets. The performance comparison metric was the area under the ROC curve (AUC).

There were no significant differences in the classification results when the input data were generated by various vectorization methods. Similarly, the classifiers yielded similar results for vectorization methods CBOW and Skip-gram. When comparing all four vectorization methods, both architectures of Doc2Vec (CBOW and Skip-gram) outperformed the other vectorization methods across all tested classification methods.

Acknowledgments

I would like to express my sincere gratitude to the Universidade Federal Fluminense (UFF) for making this work possible and for its continued commitment to academic excellence.

References

AYYADEVARA, V. K. Pro machine learning algorithms. *Apress: Berkeley, CA, USA*, Springer, 2018.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

BREIMAN, L.; FRIEDMAN, J.; STONE, C. J.; OLSHEN, R. A. *Classification and regression trees*. [S.l.]: CRC press, 1984.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 785–794.

CHEN, T.; HE, T.; BENESTY, M.; KHOTILOVICH, V.; TANG, Y.; CHO, H.; CHEN, K.; MITCHELL, R.; CANO, I.; ZHOU, T.; LI, M.; XIE, J.; LIN, M.; GENG, Y.; LI, Y.; YUAN, J. *xgboost: Extreme Gradient Boosting*. [S.l.], 2023. R package version 1.7.3.1. Disponível em: ⟨https://CRAN.R-project.org/package=xgboost⟩.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, p. 273–297, 1995.

FAWCETT, T. An introduction to roc analysis. *Pattern recognition letters*, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FEINERER, I.; HORNIK, K.; MEYER, D. Text mining infrastructure in r. *Journal of Statistical Software*, v. 25, n. 5, p. 1–54, March 2008. Disponível em: ⟨http://www.jstatsoft.org/v25/i05/⟩.

FRITSCH, S.; GUENTHER, F.; WRIGHT, M. N. *neuralnet: Training of Neural Networks*. [S.l.], 2019. R package version 1.44.2. Disponível em: ⟨https://CRAN.R-project.org/package=neuralnet⟩.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2009. v. 2.

JOSEPH, P.; YERIMA, S. Y. A comparative study of word embedding techniques for sms spam detection. In: IEEE. *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. [S.l.], 2022. p. 149–155.

KIM, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

KUBRUSLY, J.; NEVES, A. L.; MARQUES, T. L. A statistical analysis of textual e-commerce reviews using tree-based methods. *Open Journal of Statistics*, Scientific Research Publishing, v. 12, n. 3, p. 357–372, 2022.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: PMLR. *International conference on machine learning*. [S.l.], 2014. p. 1188–1196.

LIAW, A.; WIENER, M. Classification and regression by randomforest. *R News*, v. 2, n. 3, p. 18–22, 2002. Disponível em: ⟨https://CRAN.R-project.org/doc/Rnews/⟩.

LIN, X. Sentiment analysis of e-commerce customer reviews based on natural language processing. In: *Proceedings of the 2020 2nd International Conference on Big Data and Artificial Intelligence*. [S.l.: s.n.], 2020. p. 32–36.

LING, J.; CHEN, Y. Online twitter bot detection: A comparison study of vectorization and classification methods on balanced and imbalanced data. Engineering Archive, 2023.

MCCULLAGH, P. *Generalized linear models*. [S.l.]: Routledge, 2019.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, Elsevier, v. 5, n. 4, p. 1093–1113, 2014.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, v. 26, 2013.

QASEM, A. E.; SAJID, M. Exploring the effect of n-grams with bow and tf-idf representations on detecting fake news. In: *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*. [S.l.: s.n.], 2022. p. 741–746.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2019. Disponível em: ⟨https://www.R-project.org/⟩.

RINKER, T. W. *textstem: Tools for stemming and lemmatizing text*. Buffalo, New York, 2018. Version 0.1.4. Disponível em: ⟨http://github.com/trinker/textstem⟩.

ROBIN, X.; TURCK, N.; HAINARD, A.; TIBERTI, N.; LISACEK, F.; SANCHEZ, J.-C.; MüLLER, M. proc: an open-source package for r and s+ to analyze and compare roc curves. *BMC Bioinformatics*, v. 12, p. 77, 2011.

SCHÜTZE, H.; MANNING, C. D.; RAGHAVAN, P. *Introduction to information retrieval*. [S.l.]: Cambridge University Press Cambridge, 2008. v. 39.

SILGE, J.; ROBINSON, D. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, The Open Journal, v. 1, n. 3, 2016. Disponível em: ⟨http://dx.doi.org/10.21105/joss.00037⟩.

SUTTON, C. D. Classification and regression trees, bagging, and boosting. *Handbook of statistics*, Elsevier, v. 24, p. 303–329, 2005.

THERNEAU, T.; ATKINSON, B. *rpart: Recursive Partitioning and Regression Trees*. [S.l.], 2018. R package version 4.1-13. Disponível em: ⟨https://CRAN.R-project.org/package=rpart⟩.

WIJFFELS, J. *doc2vec: Distributed Representations of Sentences, Documents and Topics*. [S.l.], 2021. R package version 0.2.0. Disponível em: ⟨https://CRAN.R-project.org/package=doc2vec⟩.

_____. *word2vec: Distributed Representations of Words*. [S.l.], 2021. R package version 0.3.4. Disponível em: ⟨https://CRAN.R-project.org/package=word2vec⟩.