
GROUP_40
TEXT
MINING
REPORT 2023
AFONSO__REYNA
MARIANA_RODRIGUES
EVANS_ONORIERU
PEDRO_FERNANDES

1. INTRODUCTION

"Natural Language Processing (NLP) is the computerized approach to analyzing text that is based on both a set of theories and a set of technologies."

The field of Natural Language Processing (NLP) has gained significant attention and importance in recent years, owing to its potential in extracting valuable insights from textual data. In this project, we aim to leverage NLP techniques to predict whether a property listed on Airbnb will be unlisted in the next quarter. By analyzing real Airbnb property descriptions, host descriptions, and comments from previous guests, we will develop an NLP classification model capable of determining, for each property, whether it is likely to be unlisted (1) or still listed (0).

2. DATA EXPLORATION

The purpose of this section is to analyze and gain insights from our datasets containing property and host descriptions and guest comments. Our exploration will involve analyzing the most frequent words in each column with text information, calculating average sentence lengths, analyzing the text and target variables and visualizing the results through bar plots, pie charts, and boxplots. This analysis will provide valuable insights into the characteristics of property descriptions, host descriptions and guest comments, ultimately aiding in the development of an effective NLP classification model.

2.1. Text Variables Analysis

- Word Frequency Analysis

The analysis of the most frequent words can provide valuable insights and be used for various purposes in the project. It allowed us to determine which words were most frequent between listed and unlisted properties. Also, this analysis guided the text preprocessing steps.
After some exploration, we found out that the most frequent words in *comments* column appear significantly more frequently than in the *description* and *host_about* columns. This is obviously due to the huge magnitude of comments compared to around only 12K properties. This could indicate that the comments contain more information and contribute more to the overall data than the other ones. Additionally, we observed that for all the columns evaluated, the most frequent words were stopwords, as it is possible to see in *Figure 1.* To solve this, later as a preprocessing technique, we decided to remove common stopwords from the text, since they are unlikely to contribute much to the prediction.

- Sentence Length Analysis

Based on the average sentence length of each column, the *description* average was 814.68, meaning that on average, the property descriptions provided on Airbnb were relatively long and detailed, suggesting that hosts invest more effort in providing detailed information about

their properties. The average sentence length for the *host_about* was 439.56, indicating that on average, the host descriptions provided are moderately long. Finally, the average sentence length for *comments* was 280.58, which is a relatively shorter length, as guests tend to provide feedback based on their experiences.
Additionally, to gain some more insights, we conducted a box plot analysis. The boxplot further illustrates that the property descriptions tend to be relatively long, with the majority between 625 and 1000 characters. Moreover, regarding host about, it demonstrates a wide range of lengths. The minimum length was one, indicating the presence of at least one very short sentence. It is also possible to see that most of the host sentences fell within the range of approximately 152 and 612 characters. Finally, the boxplot analysis for comments revealed a diverse distribution of lengths, with a minimum value of 0, indicating the presence of entries with no comments or empty comments. The maximum length of 6403 indicates the presence of at least one very long comment. The boxplot shape suggests that the majority of comments were of moderate length, while the presence of outliers highlighted the presence of both very short and very long comments.

2.2. Target Variable Analysis

Based on the class distribution of the *unlisted* variable, there are 72,3% of the properties labeled as 0 (listed) and 27,7% of the properties labeled as 1 (unlisted). The class distribution is imbalanced, with a majority of the properties being listed and a small proportion being unlisted. This imbalance could potentially affect the performance and accuracy of the classification model, as the model may have a bias towards the majority class. The unlisted prediction becomes more challenging due to the limited representation of the unlisted class in the dataset. It suggests that the model may have fewer examples to learn from when predicting the unlisted properties accurately. In order to gain some more valuable insights into the relationship between the target and other variables, we decided to do a correlation matrix, concluding that *description_length* and *word_count* have a negative correlation with the target variable of approximately -0.14, which suggests that as the length of the description or the number of words in the description increases, the likelihood of the property being unlisted decreases slightly. However, the correlation is not very strong. Additionally, *host_length* and *host_word_count* have a weak positive correlation with the target of approximately 0.0067 and 0.00097, respectively. This indicates that there is a very slight tendency for properties with longer host descriptions or more host words to be listed rather than unlisted, but the correlation is very weak.

3. DATA PREPROCESSING

Data preprocessing plays a crucial role in Natural Language Processing models, impacting in a significant way their performance and accuracy. NLP tasks involve working with unstructured text data, which often requires careful preprocessing to transform the raw text into a format suitable for analysis and modeling. The main goal with this step is noise removal. It enhances the efficiency, effectiveness, quality, relevance, and consistency of text data, allowing models to extract meaningful features, improve generalization and make more accurate predictions. We found that text data contained irrelevant information, such as special characters, emojis or punctuation. By removing such noise, these preprocessing steps ensure that the model

can focus on the most relevant information and reduce the chance of learning from irrelevant information.

3.1. Language Detection - **Extra Work**

We started our preprocessing by analyzing the language distribution, we employed language identification techniques to classify each comment into its corresponding language. By applying this technique, we can determine the dominant languages present in the dataset, enabling us to focus on the top languages and discuss their implications for further analysis. Additionally, we explored potential approaches to handle multilingual data. For all text columns, the majority language is english (en), with 461793 occurrences in comments, representing 65.9% of all languages. From our 43 different languages, we only considered the first 7 with the highest percentage weight in the dataset *Figure 2* for our analysis, being these: English, French, Portuguese, Spanish, German, Italian and Dutch. The remaining 36 languages represented only 2,64% of our data. With a smaller dataset, the computational and time resources required for preprocessing are significantly reduced, leading to better model performance.

3.2. Text Cleaning

The following steps were done to reduce the noise and focus on meaningful content. Text cleaning here refers to the process of removing or transforming certain parts of the text so that the text becomes more easily understandable for NLP models that are learning the text.

- Remove special characters and emojis - **Extra Work**

Removing special characters, punctuation, and emojis from text gave us several advantages. Special characters, such as symbols or non-alphanumeric characters, can often introduce noise and disrupt the flow of the text. Removal of punctuation marks helps in simplifying the text and standardizing it for analysis. Furthermore, emojis are not always compatible with text analysis tasks. By eliminating them, we reduce the complexity of the data and focus on the essential words that carry the core information. After applying these text preprocessing techniques, we observed a decrease in the sentence length. The description initially had an average length of 827.31 characters, which was reduced to 768.50 characters after preprocessing. Similarly, the *host_about* sentences decreased from an average length of 448.71 to 424.74 characters, and the comments sentences reduced from 284.46 to 272.06 characters.

3.3. Tokenization

Our goal with Tokenization was to convert raw text data into a format that can be readily processed by algorithms. By splitting the text into individual tokens, we created a structured representation that facilitates subsequent analysis, such as feature engineering. For our tokenization process, we employed word tokenization that breaks the text into tokens based on word boundaries. This approach allowed us to obtain valuable insights.

3.4. Normalization

Normalization can help identify meaningful elements of the text and make it easier to manipulate and analyze. Lemmatization can help group words with similar meanings and reduce the size and complexity of the vocabulary. Furthermore, stop words removal can reduce the noise and redundancy of the text.

- Lower Casing

This step ensures that the same word in lower case or not, represents the same. When not converted, those two words are represented as two different words in the vector space model.

- Stopwords

After lowercase everything, we decided to remove the stopwords to reduce the dimensionality of the text data, improving computational efficiency and preventing models from overfitting. As well as we did in languages to keep, we removed stopwords only for those 7 languages kept. This decision was made based on the improvement of computational efficiency as well as it allowed us to prioritize the data that was most relevant for our analysis, since the comments for these 7 languages represented 97.36% of the entire dataset.

- Lemmatization

The main goal in this step was to extract meaningful features from text data. This facilitates the identification of important keywords, while also allowing us to return to the root form of the word, thus reducing the complexity of the dataset. After applying the lemmatization, these were the final average lengths for each variable: 82.94 for description, 42.61 for host about and 26.45 for comments.

These preprocessing steps resulted in a substantial reduction in the size of the text data. The description was reduced on average by 90%, while the host about and comments were reduced by 91%. This reduction indicates that our preprocessing techniques effectively removed unnecessary information and noise from the text, improving efficiency, and benefiting the modeling. Also, this is expected to improve the computational speed for tasks such as training models or performing analysis. To end our data preprocessing, we joined the three columns as strings by their index into one called *concat*, that represents all information about description, host about and comments after transformations.
Concluding, the preprocessing techniques applied to the dataset have successfully transformed the data into a clean data for further analysis. The removal of special characters, punctuation, emojis, stop words, and the application of tokenization, lowercasing, and lemmatization have significantly improved the quality of the data.

4. FEATURE ENGINEERING

Our goal is to first create a useful embedding for each sentence in our dataset, and then use these embeddings to accurately predict the relevant category. We experimented with two of the feature engineering techniques seen in classes: TF-IDF and Bag of Words and two transformed-based embeddings: DistilUSE transformer and Stsb transformer.

4.1. TF-IDF

One of the main advantages of TF-IDF is its ability to capture the significance of terms in a document. We decided to use TF-IDF since it provides a quantitative representation of text data, allowing algorithms to work effectively with textual data. By applying TF-IDF to our 'concat' column, we were weighting words by how frequent they are, discounting words that are too frequent, as they just add noise. In addition, we recognized the importance of including information from the language of description and host_about that were stored in two different columns and include these columns in our analysis. We encoded them using the one-hot encoding technique, transforming the categorical language information into numerical form, enabling us to add it into our model.

4.2. Bag of Words (BoW)

We decided to use Bag of words because it is simple to implement and understand. It represents text as a collection of unique words and their frequencies, making it straightforward to compute and interpret. It also has an advantage for us, which is language-independence, meaning it can be applied to texts in any language and we are working with 7.

4.3. Transformed-based embeddings - **Extra Work**

Unlike traditional word embeddings, transformer-based embeddings generate contextualized representations. They consider the entire context of a word within a sentence, which allows for more accurate encoding of words.

- DistilUSE transformer

DistilUSE is a version of the Universal Sentence Encoder (USE) model, providing comparable performance to the full USE model but with a smaller size. It is a Multi-Lingual model of Universal Sentence Encoder for 50 languages. DistilUSE is trained using unsupervised learning and it consists of multiple layers of self-attention and feed-forward neural networks. Self-attention allows the model to weigh the importance of each word in the context of the entire sentence, being able to capture long-range dependencies and relationships between words. After that, mean pooling and max pooling were applied, to aggregate the representations of individual words into a fixed-length vector representation for the entire sentence.

- Stsb-xlm-r-multilingual

STSB-XLM-R-Multilingual is also specifically designed to handle multiple languages. This embedding can capture language-agnostic features, making them useful in multilingual NLP applications where understanding and processing text from different languages is required. It breaks down sentences into smaller units, represents them as embeddings and utilizes a transformer encoder, allowing the model to capture contextual relationships and dependencies between words. It randomly masks out some words in the input sentences and trains the model to predict the masked tokens based on their surrounding context. After pre-training, this transformer can be fine-tuned on specific datasets, enabling it to adapt to the specific task.

5. CLASSIFICATION MODELS

After implementing and experimenting two embedding methods and then two extra methods using pre-trained transformer-based embeddings, we can now implement, and test three classification models seen in class and two more advanced models.

5.1. K-Nearest Neighbors, Logistic Regression, MLP

Our KNN algorithm considers the 10 nearest neighbors to make predictions. We chose a smaller K value to a more localized prediction. The parameter *weights* defines that the distance between the neighbors and the target data point is used as a weight to determine the contribution of each neighbor in the prediction. We decided to use Logistic Regression because it is a popular algorithm for binary classification. The MLP is a type of artificial neural network with multiple layers of interconnected nodes. It is capable of learning complex relationships between inputs and outputs, making it suitable for various classification tasks.

5.2. More advanced models - **Extra Work**:

We decided to use XGBoost, because it is an advanced gradient boosting algorithm known for its efficiency and high performance. Random Forest was chosen because it aims to achieve better generalization and robustness. This algorithm is also known for its ability to handle high-dimensional data and noisy features.

6. EVALUATION

We implemented KNN, LR, MLP, XGB and RF for all models, we compared them and evaluated their performance using accuracy, recall, precision, and F1-score, gaining a comprehensive understanding of how each model performs in different scenarios. With this, we will be able to make informed decisions for our specific task.

		TF-IDF	BOW	DistilUSE	stsb
KNN	Accuracy	0,75	0,76	0,8	0,79
	Precision	0,21	0,27	0,43	0,43
	Recall	0,55	0,6	0,7	0,67
	F1-score	0,31	0,37	0,54	0,52
Logistic	Accuracy	0,76	0,72	0,75	0,76
	Precision	0,13	0,43	0,11	0,31
	Recall	0,75	0,46	0,71	0,58
	F1-score	0,22	0,45	0,19	0,41
MLP	Accuracy	0,73	0,74	0,77	0,76
	Precision	0,4	0,4	0,55	0,49
	Recall	0,49	0,51	0,57	0,55
	F1-score	0,44	0,45	0,56	0,52
XGB	Accuracy	0,76	0,77	0,79	0,79
	Precision	0,28	0,27	0,4	0,38
	Recall	0,62	0,62	0,66	0,69
	F1-score	0,38	0,38	0,5	0,49
RF	Accuracy	0,77	0,77	0,8	0,8
	Precision	0,13	0,144	0,29	0,3
	Recall	0,89	0,88	0,84	0,83
	F1-score	0,22	0,23	0,43	0,44

Among the classification algorithms, MLP and KNN achieved an F1 score of 56% for and 54%, both for DistulUSE. However, KNN outperformed MLP in terms of accuracy and recall, achieving values of 80% and 70%, respectively. Regarding TF-IDF and BoW, XGB and Random Forest achieved the biggest Accuracy of 77% in both, but XGB outperformed in terms of F1-score, with values of 38% for both. Ultimately, our evaluation revealed that the DistilUSE embedding with MLP classifier emerged as the top-performing model across all metrics. This model achieved an F1 score of 56%, precision of 55%, recall of 57%, and accuracy of 77%.
In conclusion, the DistilUSE embedding with MLP classifier stands as the most effective model for predicting whether a property listed on Airbnb will be unlisted.

7. HYPERTUNING

To optimize the performance of our MLP model with DistilUSE embeddings, a grid search was conducted over a predefined parameter grid. The goal was to identify the best combination of hyperparameters that maximizes the F1 score. The following hyperparameters were explored:

- Hidden Layer Sizes: We considered various configurations of the hidden layer sizes for our grid search, including (50,), (100,), (50, 50), and (100, 50). These combinations determine the number of neurons in each hidden layer.
- Activation Function: Two activation functions, 'relu' and 'tanh', were tested to observe their impact on the model's performance. The 'relu' function introduces non-linearity by thresholding any negative input to zero, while the 'tanh' function maps the input to values between -1 and 1.
- Alpha (Regularization Parameter): 3 values of alpha, namely 0.0001, 0.001, and 0.01, were examined. This parameter controls the amount of regularization applied to the weights, preventing overfitting.
- Learning Rate: The 'adaptive' learning rate schedule was chosen to adjust the learning rate for each update based on the training loss.

This configuration yielded a F1 score of 0.5596 on the training data. It should be noted that the reported score is obtained through cross-validation, providing a robust estimate of the model's performance.

8. READY-TO-RUN

To apply our trained model on the test data and make predictions, we need first to ensure that the test data undergoes the same preprocessing steps as the training data. Similarly to the training data, language detection is performed to identify the language of each text sample and to keep the same 7 that we keep in the training data. The detected language information is then joined to the test dataframe for further analysis and language-specific preprocessing steps. Special characters, emojis, and punctuation marks are also removed from the test, eliminating noise and unnecessary information. The test data is tokenized by splitting the text samples into individual words. All tokens in the test data are converted to lowercase to standardize the text and avoid duplicating representations of the same word. After that, stopwords are removed from the test data and lemmatization is applied to transform words into their base or dictionary form. After preprocessing each text sample, the individual pieces are joined back together and concatenated to create a unified dataset ready for modeling. To effectively represent the test data, we need to determine the best embedding method. The best model, as determined through the hyperparameter tuning process, is trained on the whole preprocessed and embedded training data. The preprocessed and embedded training data is imported to ensure consistency in the feature engineering and encoding steps. This allows the model to operate on the same feature space as it did during training. The trained model is then applied to the preprocessed test data to make predictions. By feeding the preprocessed test data into the model, we can generate predictions for the respective text samples.

8.1. Final Predictions

The description and host_about columns, which were not necessary for the predictions analysis, were dropped from the merged dataset. In some cases, the model did not predict since we removed some of the languages in the test data. To solve this, missing predictions were filled using a random selection method. The ratio between the counts of predicted classes (1s and 0s) in the available predictions was calculated. Then, for the missing predictions, a random choice was made between the two classes (0 and 1) based on this ratio. This approach ensures that the distribution of predicted classes remains consistent with the available predictions. The counts of each predicted class (0s and 1s) were calculated and analyzed to gain insights into the distribution of the predictions. The final predictions, along with the corresponding text samples and other relevant information, were saved in a CSV file named "Predictions_40.csv". By following this process, we were able to generate predictions for the test data using our trained model. The resulting predictions and their associated information provide valuable insights into the classification of the text samples and contribute to the overall analysis and conclusions of the project.

1. ANNEXES

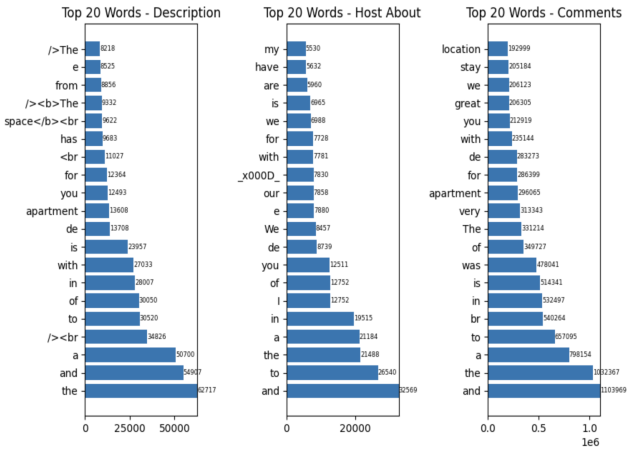

Figure 1: Top 20 words for description, host_about and comments

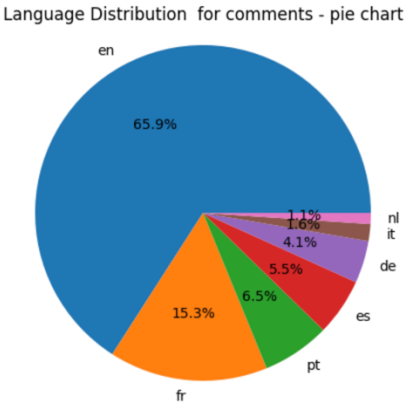

Figure 2: languages %

2. BIBLIOGRAPHY

MBCS, D. A. (2023). *A Comprehensive Guide To Feature Engineering with N-Grams For Natural Language Processing*. Obtido de Linkedin: https://www.linkedin.com/pulse/comprehensive-guide-feature-engineering-n-grams-david-adamson-mbcs/

CORNEL, S. (2020). *NLP Text Mining - Disease behaviour*. Obtido de Kaggle: https://www.kaggle.com/code/cstefanache/nlp-text-mining-disease-behaviour

Liddy, E. D. (s.d.). *Natural Language Processing.* Obtido de Syracuse University: https://surface.syr.edu/cgi/viewcontent.cgi?article=1043&=&context=istpub&=&sei-redir=1&referer=https%253A%252F%252Fscholar.google.com.br%252Fscholar%253Fhl%253Den%2526as_sdt%253D0%25252C5%2526q%253Dnatural%252Blanguage%252Bprocessing%2526btnG%253D%2526oq

What are the best tools and techniques for text normalization and standardization? (s.d.). Obtido de Linkedin: https://www.linkedin.com/advice/1/what-best-tools-techniques-text-normalization?trackingId=yP1b88%2B1PkQ%2BWgK6EtYLvg%3D%3D&lipi=urn%3Ali%3Apage%3Ad_flagship3_showcase%3Bis3Vx3%2B2ShGapfmKqI6rTw%3D%3D

Treviso, M. (s.d.). *Efficient Methods for Natural Language Processing: A Survey.*

How do you optimize the number of topics and the hyperparameters of your topic modeling algorithm? (s.d.). Obtido de linkedin: https://www.linkedin.com/advice/1/how-do-you-optimize-number-topics-hyperparameters?trackingId=PJ%2FWi4scFmwUMKis%2BPKCbA%3D%3D&lipi=urn%3Ali%3Apage%3Ad_flagship3_showcase%3Bis3Vx3%2B2ShGapfmKqI6rTw%3D%3D

How do you evaluate the accuracy and efficiency of lemmatization tools? (s.d.). Obtido de linkedin: https://www.linkedin.com/advice/3/how-do-you-evaluate-accuracy-efficiency-1e?trackingId=zDDQbJpyOup3cDMliOnZfQ%3D%3D&lipi=urn%3Ali%3Apage%3Ad_flagship3_showcase%3Bis3Vx3%2B2ShGapfmKqI6rTw%3D%3D

How do you design and implement NLP pipelines and workflows? (s.d.). Obtido de Linkedin: https://www.linkedin.com/advice/0/how-do-you-design-implement-nlp-pipelines?trackingId=wSXJP5ggX5XS78DlcPj38w%3D%3D&lipi=urn%3Ali%3Apage%3Ad_flagship3_showcase%3Bis3Vx3%2B2ShGapfmKqI6rTw%3D%3D

Hassan Badawy. (23 de feb de 2023). *BOW vs TF-IDF for NLP Text Vectorization*. Obtido de Linkedin: https://www.linkedin.com/pulse/bow-vs-tf-idf-nlp-text-vectorization-hassan-badawy-msc/

Badawy, H. (23 de feb de 2023). *BOW vs TF-IDF for NLP Text Vectorization*. Obtido de linkedin: https://www.linkedin.com/pulse/bow-vs-tf-idf-nlp-text-vectorization-hassan-badawy-msc/

sentence-transformers/distiluse-base-multilingual-cased-v2. (s.d.). Obtido de Hugging face: https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2

Higging Face. (s.d.). Obtido de sentence-transformers/stsb-xlm-r-multilingual: https://huggingface.co/sentence-transformers/stsb-xlm-r-multilingual