PREDICTING AIRBNB UNLISTING

Group 12

Project developed by:

Ayotunde Aribo, 20221012

Ehis Jegbefumwen, 20221015

Oseias Beu, 20230524

Lizaveta Barysionak, 20220667

TABLE OF CONTENTS

1. INTRODUCTION

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 Airbnb property descriptions, host descriptions, and guest comments, we will develop an NLP classification model to determine whether each property is likely to be unlisted (1) or remain listed (0).

2. DATA EXPLORATION

In this section, we aim to gain insights from our datasets, which include property descriptions, host descriptions, and guest comments. Our exploration involves analysing the most frequent words in each text column, calculating average sentence lengths, examining the relationship between the text and target variables, and visualising the results using a Pie chart and word cloud. Additionally, we will check for and address any null values. This analysis will provide a comprehensive understanding of the data's characteristics, helping in the development of an effective NLP classification model.

2.1. INITIAL SETUP AND DATA LOADING

To begin with, we ensured all necessary packages were installed and loaded. This includes libraries for text processing, data manipulation, and visualisation such as Pandas, matplotlib and word cloud

2.2. DATA OVERVIEW

We loaded the training datasets (train_df and train_reviews_df) and performed an initial inspection. We first displayed the first few rows of each dataset and checked their shape and column names. Specifically, the train dataset has a shape of (6248, 4) and the train reviews dataset has a shape of (361281, 2). We then identified and addressed any null values in the datasets. In train_df, there were no null values across the columns (index, description, host_about, unlisted). However, in train_reviews_df, there were 2 null values in the comments column. Sample data from key columns such as property descriptions and host descriptions were inspected to understand their content better.

2.3. TARGET VARIABLE DISTRIBUTION

To understand the class imbalance, we analysed the distribution of the target variable (unlisted). We visualised this distribution using a pie chart, which highlighted the proportion of properties that were listed versus unlisted. The pie chart revealed that a significant portion of the properties were listed (72.7%), with about a third of the fraction being unlisted (27.3%). This visualisation provided clear insight into the class distribution, indicating a notable imbalance in the dataset. The imbalance in the dataset can impact the performance of our predictive models, as they may exhibit bias towards the majority class. This could lead to suboptimal predictions for the minority class, affecting the overall accuracy and reliability of our analyses.

2.4. TEXT DATA ANALYSIS

To gain deeper insights into the textual data, we performed the following analyses:

- **Word Frequency Analysis**

The analysis of word frequency offers valuable insights for our text mining project, guiding our understanding of listed and unlisted properties on Airbnb. Notably, comments contain richer information compared to descriptions and host_about sections, suggesting their significance in the dataset. Our examination revealed that common stop words dominate across all columns, indicating their pervasive presence and potential hindrance to prediction accuracy. Interestingly, among the most frequent words in the "Host about" section are 'and' and 'to', indicating the conversational nature of the content. Similarly, in the "Property description" field, 'the' and 'and' are prevalent, highlighting the narrative style of property listings. In reviews, common words such as 'the' and 'and' are prominent, emphasising the ubiquitous nature of feedback language. Sample words from reviews include 'great' and 'apartment', reflecting sentiments and topics often discussed. To address this, we will implement some preprocessing techniques to remove these stop words and refine our analysis. This strategic approach enhances the quality and accuracy of our predictive models, enabling more robust insights and informed decision-making.

- **Sentence Length Analysis**

The average sentence length for property descriptions on Airbnb was 81,468 characters, suggesting hosts invest considerable effort in providing detailed information. Host descriptions averaged 43,956 characters, indicating moderate length. Guest comments averaged 280.58 characters, reflecting shorter feedback based on experiences.

To gain further insights, we analyzed the correlation matrix of the numeric variables, including description length, description word count, host about length, host about word count, and the target variable unlisted. The matrix revealed a strong positive correlation between description length and description word count (0.96), as well as between host about length and host about word count (0.99). However, the correlations between these variables and the target variable unlisted were relatively weak, with the highest being -0.13 for description length and unlisted. This suggests that while the lengths and word counts of descriptions and host information are related to each other, they have a minimal direct correlation with whether a property is unlisted.

3. PREPROCESSING

3.1. BEFORE EXPLORATION

Before carrying out further data analysis, we carried out basic preprocessing on both the training and testing datasets to maintain data consistency in any analysis done. We initially

checked for null values and there were none on both datasets as shown in Image 1. Then we performed the following steps were performed:

HTML Tag Removal – Extra HTML tags can interfere with NLP models. A function was implemented to remove HTML tags using regular expressions, ensuring the text data is free from HTML artifacts.

Lowercasing To standardize text and avoid case-sensitivity issues, a function was applied to convert all text to lowercase, reducing text variability and ensuring uniformity. We combined the above functions into a single function (Clean Texts) that sequentially applies lowercasing and HTML tag removal. This integrated function ensures that each text column undergoes both preprocessing steps, resulting in consistently clean text data. 269,682 tags were removed from train reviews and 31,162 were removed from test reviews

3.2. AFTER EXPLORATION

Language Detection – Extra

We began preprocessing by analyzing the language distribution using language identification techniques to classify each comment. This allowed us to identify the dominant languages in the dataset, enabling us to focus on the top languages and discuss their implications for further analysis. We explored potential approaches for handling multilingual data.

For all text columns, the majority language was English, with 231,216 occurrences, representing 64.2% of all languages. Out of 43 unique languages, we considered only the top 10, which constituted 98.22% of the dataset [Figure 2]. These languages were English, French, Portuguese, Spanish, German, Italian, Dutch, Russian, Korean, and Romanian. The remaining 33 languages accounted for only 1.88% of the data. By focusing on a smaller subset, we significantly reduced the computational and time resources required for preprocessing, leading to improved model performance.

Replace Emojis with Characters - Extra

We imported the emoji library to process emojis in the reviews dataset. Our goal was to convert emoji codes into their corresponding characters rather than removing them. Among the reviews, 177 of the training reviews contained only emojis.

Tokenization

We then split the text with tokenization, into meaningful units (tokens) that represent the smallest semantic components of the text. A total of 21,880331 from the train and test review datasets.

Punctuation

We applied a punctuation function to several columns in our training and testing datasets to replace non-alphanumeric characters with spaces and then remove empty strings and spaces. The number of punctuations removed from each column is as follows:

- `train_df['description']`: 206,982
- `train_df['host_about']`: 169,401
- `train_reviews_df['comments']`: 4,706,744
- `test_df['description']`: 23,162
- `test_df['host_about']`: 18,066
- `test_reviews_df['comments']`: 537,083

Remove non-word strings – Extra

We added a further step to clean up the data and remove non-alphabetic tokens. The function was applied to the same columns in the punctuation function

Remove stopwords

Given the multilingual nature of our dataset, we downloaded stopwords for all relevant languages, not just English. We then identified and removed the total number of stopwords in the text columns across the train and test datasets, which amounted to 10,889,540.

Lemmatize comments

Finally, we developed a custom function, `lemmatize_comments`, to perform lemmatization on tokenized text data. After applying the lemmatization function, the text data in the specified columns was normalized to its base form.

The preprocessing steps we used significantly improved data quality and reduced the dataset's text length. This enhancement in data quality is crucial for the accuracy and efficiency of our NLP model. After finishing the preprocessing, we merged the description dataset with the reviews dataset and combined all the text columns into one called 'combined_text'. This way, we could use all the relevant details about the properties to train our model, especially for those without reviews. Then, we carried out feature engineering using this combined text column.

4. FEATURE ENGINEERING

Our goal is to create effective embeddings for each sentence in our dataset and use these embeddings to accurately predict. We experimented with two feature engineering techniques covered in class: GloVe Embedding and TF-IDF, as well as a transformer-based embedding: DistilBERT transformer.

4.1. GLOVE EMBEDDING

For the feature engineering process, we utilized GloVe (Global Vectors for Word Representation) embeddings to convert textual data into numerical format, which is essential for training machine learning models. The following steps outline the methods and parameters used in this process. We used pre-trained GloVe embeddings (glove.6B.300d.txt), consisting of 300-dimensional vectors. The embeddings were loaded into a dictionary,

mapping each word to its corresponding vector representation, with an embedding dimension of 300. We tokenized the text data using the Tokenizer class from Keras. The tokenizer was fitted on the text data to convert each word into a unique integer index, with parameters including the text data to be tokenized and the maximum sequence length to pad or truncate. To ensure uniform sequence length, we padded the sequences to the maximum sequence length found in the data. This step is crucial for maintaining consistent input dimensions for the model. An embedding matrix was created where each row corresponds to a word's vector representation. Words not found in the GloVe embeddings were initialized with zeros.

Parameter: embedding_matrix (a matrix of shape (len(word_index) + embedding_dim)).

By implementing these methods, we effectively transformed the raw text data into a numerical format suitable for model training, leveraging the semantic information from pre-trained GloVe embeddings to enhance the model's performance.

4.2. TF-IDF (TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY)

TF-IDF measures the importance of a word in a document relative to a corpus. It combines Term Frequency (TF) with Inverse Document Frequency (IDF). The TF-IDF score highlights significant words in specific documents while downplaying common words. The score is the product of TF and IDF.

To implement TF-IDF, we followed a series of steps. First, we prepared the text data by extracting it from the ***'combined_text'*** column of our dataset. We then performed TF-IDF vectorization by initializing aTfidVectorizerwith a maximum feature limit of 5000 to convert the text data into a TF-IDF matrix. This matrix represents the TF-IDF scores of words across the text data. Then, we converted the TF-IDF matrix into a DataFrame for better readability. We inspected the shape of the TF-IDF matrix and displayed the first 10 TF-IDF features to understand the data better.

Finally, we conducted feature importance analysis by summing the TF-IDF values for each feature (word) and creating a DataFrame with the words and their corresponding TF-IDF scores. We identified and visualized the top 20 words with the highest TF-IDF scores using a bar plot to highlight the most important words in the dataset.

4.3. DISTILBERT – EXTRA WORK

We employed the DistilBERT model, leveraging its tokenizer and pre-trained weights ('distilbert-base-uncased') from the Hugging Face library, to transform textual data into numerical embeddings for downstream NLP tasks. The process involved tokenizing text inputs, with parameters set for truncation and padding to a maximum length of 512 tokens, ensuring uniform input dimensions. Subsequently, we fed the tokenized inputs into the DistilBERT model to extract hidden state representations. The mean of these hidden states across the sequence length dimension was computed to generate fixed-size embeddings for each text

instance. This methodology was applied to the ***'combined_text'*** column of our training dataset, resulting in embeddings with a dimensionality of 6248 samples by 768 features. These embeddings capture the semantic nuances of the text data, facilitating their use in various analytical and predictive modeling tasks. The successful extraction and dimensional consistency of the embeddings were confirmed by verifying the output shape.

4.4. DISTILUSE – EXTRA WORK

The DistilUSE (Universal Sentence Encoder) model is employed to generate sentence embeddings, encapsulating the semantic essence of text data. Below delineates the implementation process.

We utilized the DistilUSE model (distiluse-base-multilingual-cased-v1) for embedding generation from textual data. Textual data from train_df and test_df's ***'combined_text'*** column were transformed into numpy arrays for compatibility. Using the encode method, embeddings were generated for both datasets with a progress bar for visualization. Resultant embeddings were converted into pandas DataFrames for readability and analysis. We provided an overview, including dimensions and previews of the first 5 rows, showcasing the embeddings' structure.

The DistilUSE model used is multilingual, trained on cased data, producing 512-dimensional embeddings. This streamlined approach transforms textual data into concise vectors, aiding tasks for classification.

5. CLASSIFICATION MODELS

After preprocessing the text data, our next step was to develop and evaluate different classification models to predict if an Airbnb property would be unlisted in the next quarter. We created three models using three different transformed datasets: TFIDF, DistilBert, and DistilUse. We then compared how well each model performed with each transformation to find out which one worked best. The classification models we used were Support Vector Machine (SVM), Random Forest (RF), and Logistic Regression (LR). The respective transformed files were loaded before running the classification models

5.1. FEATURE SELECTION

Metadata Features: We selected the following metadata features for general information about the properties and their hosts:

- description_length: Length of the property description.
- description_word_count: Word count of the property description.
- host_about_length: Length of the host's "about" section.
- host_about_word_count: Word count of the host's "about" section.
- description_lang: Language of the property description.
- host_about_lang: Language of the host's "about" section.

Transformed features: We included the transformed features from the embeddings model, capturing semantic information from the text. These features were extracted from the 15th column till the last column regardless of the transformed file used.

Data Preprocessing:

- Categorical variables (description_lang, host_about_lang) were one-hot encoded.
- One-hot encoded metadata features were combined with transformed features to create a comprehensive feature set.
- The dataset was split into features (X) and the target variable (y), indicating whether a property would be unlisted in the next quarter.
- Features were standardized to have a mean of zero and a standard deviation of one.
- Train- The dataset was split into training (80%) and testing (20%) sets with a random state of 42 for reproducibility.

We saved the preprocessing steps as **'encoded_columns.pkl'** and **'scaler.pkl'** so we could load it when preprocessing the test data to ensure consistency in features for train and test data

6. EVALUATION

We evaluated these models using three different transformation techniques: TFIDF, DistilBERT, and DistilUSE. Here we present the performance metrics of each model under these transformation methods and identifies the best performing model based on these evaluations.

6.1. MODEL COMPARISON AND BEST MODEL SELECTION

Model	Transformation	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Class 0)	F1-Score (Class 1)	Accuracy
SVM	TFIDF	0.93	0.74	0.89	0.82	0.91	0.78	0.87
Random Forest	TFIDF	0.92	0.74	0.89	0.8	0.91	0.77	0.87
Logistic Regression	TFIDF	0.9	0.73	0.9	0.73	0.9	0.73	0.85
SVM	DistilBERT	0.93	0.74	0.89	0.82	0.91	0.78	0.87
Random Forest	DistilBERT	0.92	0.74	0.89	0.8	0.91	0.77	0.87
Logistic Regression	DistilBERT	0.9	0.73	0.9	0.73	0.9	0.73	0.85
SVM	DistilUSE	0.81	0.8	0.96	0.4	0.88	0.53	0.81
Random Forest	DistilUSE	0.8	0.77	0.96	0.37	0.87	0.5	0.8
Logistic Regression	DistilUSE	0.81	0.65	0.91	0.46	0.86	0.54	0.78

6.1.1. Accuracy

Across all feature extraction techniques, SVM and Random Forest achieved the highest accuracy of 0.87 with both TFIDF and DistilBERT, while Logistic Regression lagged slightly behind with an accuracy of 0.85 for these methods. When using DistilUSE, the performance of all models dropped, with SVM and Random Forest achieving 0.81 and 0.80 accuracy, respectively, and Logistic Regression achieving 0.78.

6.1.2. Precision, Recall, and F1-Score

- **TFIDF and DistilBERT**: For both feature extraction methods, SVM and Random Forest displayed similar performance, with slight variations in precision, recall, and F1-score, favoring SVM slightly due to its higher recall for class 1.
- **DistilUSE**: SVM showed a significant drop in recall for class 1 (0.40), impacting its F1-score. Similarly, Random Forest and Logistic Regression showed decreased performance with lower recall and F1-scores for class 1.

6.1.3. Consistency and Robustness

SVM consistently performed well across TFIDF and DistilBERT, maintaining high precision, recall, and F1-scores, especially for class 1. Random Forest also performed well, but with slightly lower metrics for class 1 compared to SVM.

6.2. BEST MODEL

After evaluating the performance metrics of Logistic Regression, SVM, and Random Forest across different feature extraction methods, we concluded that **SVM with TFIDF** and **SVM with DistilBERT** are the best performing models. Both configurations achieved an accuracy of 0.87 and demonstrated strong performance in terms of precision, recall, and F1-score, particularly for class 1.

Given the consistency and robustness of SVM with both TFIDF and DistilBERT, we decided to use **SVM with DISTILBERT** as the final model due to its slightly better recall and F1-score for class 1, making it more effective in identifying properties that are likely to be listed on Airbnb.

After selecting the optimal model, it was saved as **'best_svm_model.pkl'**. This model, along with the saved preprocessing steps, was applied to the DISTILBERT-transformed test data to generate a CSV file of the predictions. This file is included in the zip archive accompanying this report.

Note: We attempted LSTM but the model took too much time to run and the model's performance was not encouraging so we discarded it.

7. ANNEXES

Figure 1 - Null Values on Train & Test Data

```
Dataset train_reviews shape before dropping null value : (361281, 2)
Dataset train_reviews shape after dropping null value : (361281, 2)
Dataset train shape before dropping null value : (6248, 4)
Dataset train shape after dropping null value : (6248, 4)
```

```
[9]  ✓ 0.1s
```

```
Dataset test_reviews shape before dropping null value : (41866, 2)
Dataset test_reviews shape after dropping null value : (41866, 2)
```

Figure 2 - Top Language reviews

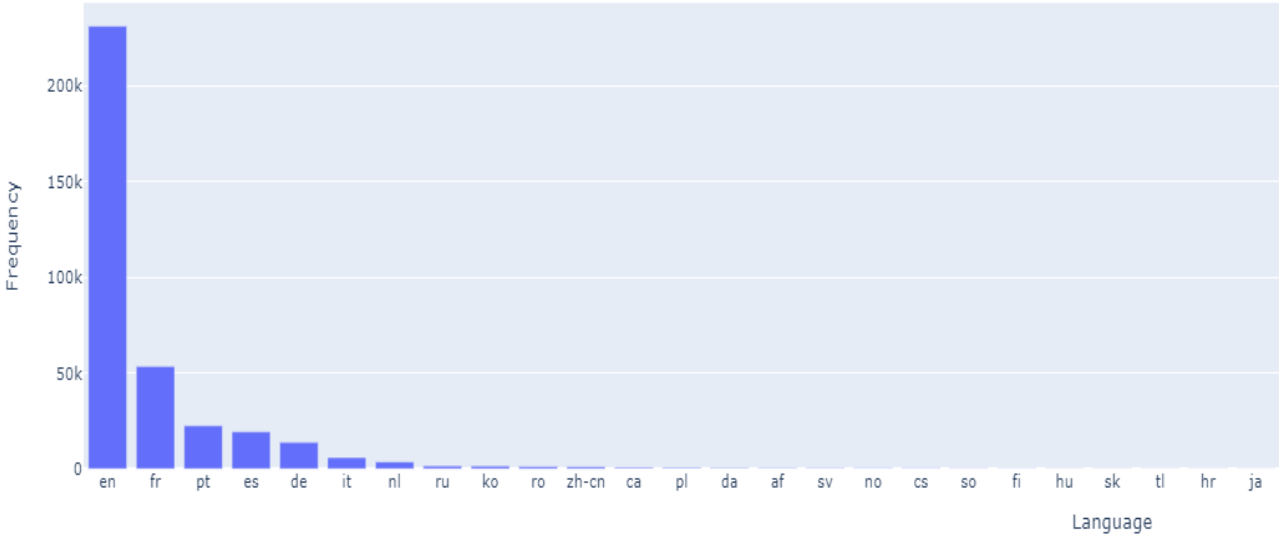


Figure 3 - Distribution of Listed and Non-Listed

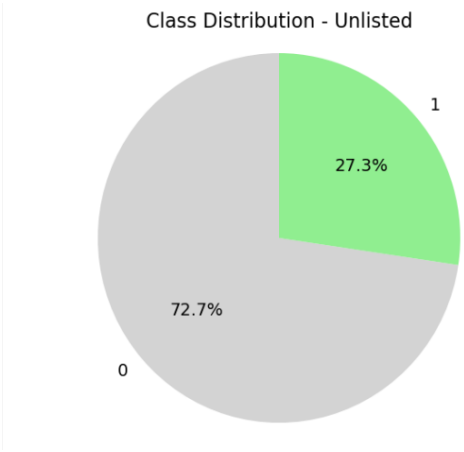


Figure 4 – Wordcloud

Figure 5 - Top Non-Stopwords

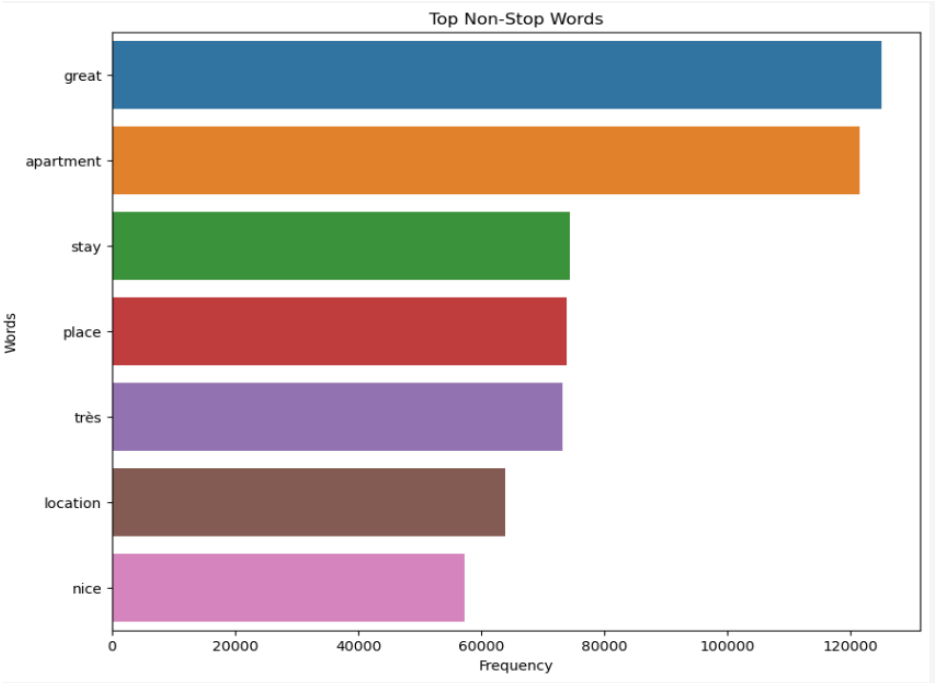
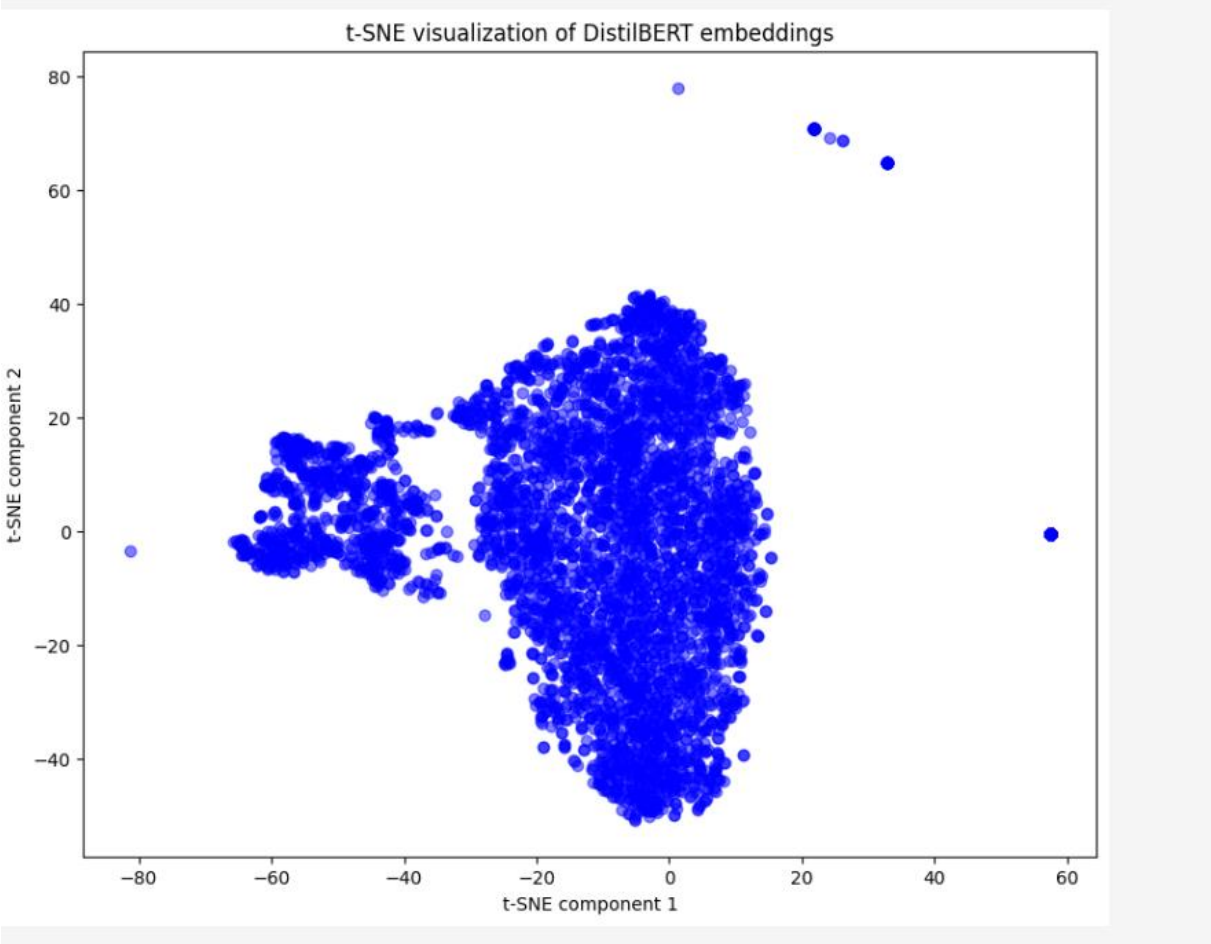


Figure 6: t-SNE Visualization – Train



8. REFERENCES

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

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa