UNIVERSIDAD CARLOS III DE MADRID

DATA SCIENCE AND ENGINEERING

Machine Learning Applications

ANALYSIS OF SPANISH POLITICAL TWEETS

Authors:

Darío Cabezas Larese

Lucía Cordero Sánchez

Lucía León Marco

Date: 7th May, 2023

Contents

1 Introduction

In this project, our primary objective was to investigate the application of Natural Language Processing (NLP) techniques for analyzing Twitter data. We employed web scraping methods to collect relevant data from Twitter and conducted several preprocessing steps to ensure the data was ready for analysis. Our overarching aim was to develop a robust model capable of accurately predicting the sentiment expressed in the tweets within our dataset. By achieving this goal, we sought to uncover meaningful patterns and trends inherent in the data.

To accomplish our objectives, we implemented a diverse set of NLP techniques along with text preprocessing, vectorization, and classification for regression tasks. In order to improve the performance of our model when applied to the scraped tweets, we conducted training using a separate dataset from Kaggle (Sentiment140 dataset with 1.6 million tweets) that featured labeled tweets classified into positive, negative, or neutral sentiment categories.

Furthermore, we delved into the utilization of pre-trained transformers to identify both positive sentiment and hate speech within the Twitter data for each political party. Additionally, we provided visualizations to facilitate the identification of tweets with similar topics. To showcase our findings effectively, we implemented a dashboard featuring graphs and other visual representations.

Collectively, our project serves as a valuable illustration of how NLP techniques can be employed to extract meaningful insights from vast volumes of unstructured text data. By employing these techniques, decision-makers across various domains, including business, social science, and public policy, can be better informed in their decision-making processes.

2 Text preprocessing and vectorization

2.1 Text preprocessing

For generating the complete data, we scrapped tweets for each of the five political parties that we selected. Then, merge them into a single data frame.

The first thing to do is to perform an exploratory analysis of the data that we already have with the labels (positive, negative, and neutral), whose source is Kaggle (more information provided later in the report). For doing the preprocessing we start by selecting just the target and the text of the tweet, so the metadata will not be used for the moment. We select just 40,000 observations so that the computation does not take so much time, by concatenating the 20,000 positive observations with the negative ones.

The preprocessing steps involved in preparing the data for analysis include:

- Converting all text to lowercase.
- Removing stopwords and punctuation marks.
- Repeating character removal.
- Removing links and numbers from the tweets.
- Tokenizing the text using the *RegexpTokenizer* function from the *nltk* package.

- Applying stemming using the Porter-Stemmer algorithm.
- Applying lemmatization using the Word Net Lemmatizer and part of speech (POS) tagging.

The*RegexpTokenizer* function takes a regular expression pattern as input and uses it to identify the boundaries between tokens in the input text. By the way, we use more functions of the package for applying to stem and lemmatize.
For stemming, we used the Porter-Stemmer algorithm, which takes a word and reduces it to its base or root form. takes a word and reduces it to its base or root form. And then applying the Word Net Lemmatizer we get a set of words in their base or dictionary form. It uses the Word Net lexical database to determine the lemma of a word. It also uses the part of speech (POS) of the word to determine the correct lemma, as different POS may have different lemmas.

Similarly, for the data collected from Twitter (the testing set), the following preprocessing steps were performed:

- Balancing the dataset by selecting 2,000 observations for each political party. We checked it out with a pie graph.
- Removing special characters, punctuations, and single characters.
- Substituting multiple spaces with a single space.
- Converting all tweets to lowercase.
- Removing Spanish stopwords using the SnowballStemmer algorithm.

Using the World Cloud module we can plot the more frequent words used in the tweets of each political party.

As it can be checked in the world-cloud graphs at this moment the frequent words are part of the Spanish set of stopwords (which is coherent but insightless) so we have to remove them to get some more significant results. So using nltk package we define a function for preprocessing the data. We use the SnowballStemmer that is based on the Snowball stemming algorithm, which is a language-specific implementation of the Porter stemming algorithm that is the one we used in the training set. In the function we removed URLs, tokenize the tweets, remove the words that are in the set of Spanish stopwords and apply the stemmer.

Finally, we pass the data through that function and get the preprocessed data. Use again the world-cloud plots for looking at the frequent words and check that we deleted the stopwords (see Figure 1).

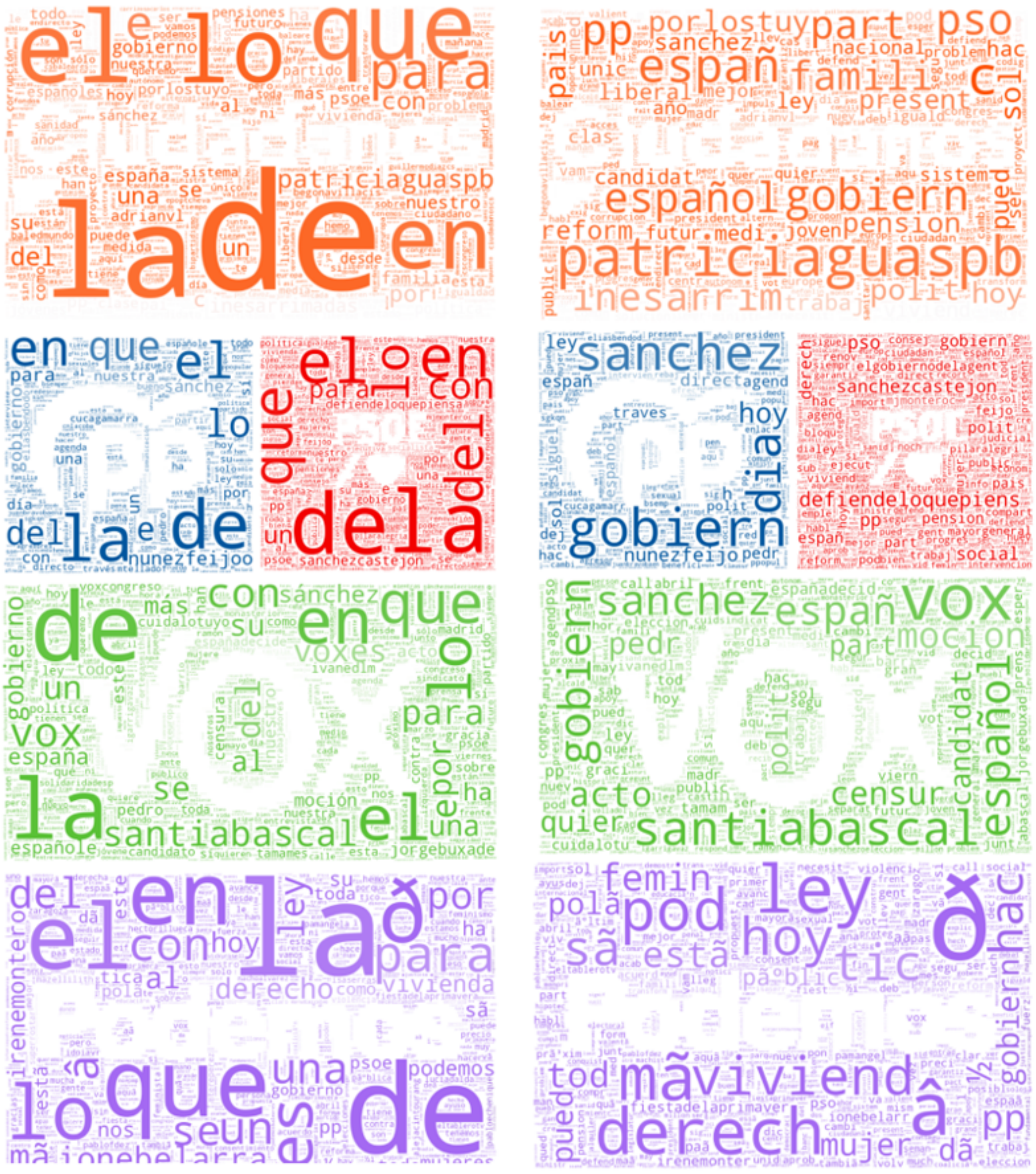
2.2 Vectorization

Vectorization uses some packages such as *sklearn* for text and *gensim* for converting the words into vectors.

Defining a function for vectorizing the text data of each observation we perform the bag of words using the *CountVectorizer* method of sklearn package, it transforms a collection of text documents into a matrix of token counts. It counts the frequency of each word in a given document and generates a sparse matrix of these counts.

We also implement the Term Frequency - Inverse Document Frequency (TF-IDF) with the *TfidfVectorizer* of the same package. Using the Term Frequency-Inverse Document Frequency technique, we can assign weights to each word vector based on its frequency in the document and inverse frequency across the entire corpus. These weighted vectors are then combined using methods like averaging or summation.

Figure 1: Left: Initial WordCloud. Right: Preprocessed Wordcloud



And finally use the *word2vec* function to convert the words into vectors, it is used for generating word embeddings.

Word embeddings capture semantic and syntactic relationships between words by representing them as dense vectors in a high-dimensional space. The main idea behind Word2Vec is to train a neural network to predict the probability of a word based on its neighboring words. During the training process, the network adjusts the word vectors to maximize the predictive accuracy. As a result, words with similar meanings or contexts tend to have similar vector representations in the embedding space.

The Word2Vec model provides a function that takes a word as input and returns its corresponding vector representation.

Configuration of feature matrix	**Approximate Accuracy**
Bag of Words (BoW)	0.735
TF-IDF	0.732
Keras Embedding	0.493
Doc2Vec	0.507

This allows us to represent each word in a text document as a vector.

Once we have the word vectors, we can apply them for vectorization in both the training set and the testing set. This vectorization process involves replacing each word in the documents with its corresponding word vector. By doing so, we transform the text data into a numerical representation that can be processed by machine learning algorithms.

It is important to note that the vectorization process should be consistent between the training set (which includes labeled data, in this case the Kaggle dataset) and the testing set (our own scrapping). This ensures that the same word vectors are used for corresponding words in both sets, enabling fair and accurate evaluation of the model's performance.

By utilizing Word2Vec-based vectorization, we can capture the semantic meanings and relationships of words in a high-dimensional vector space. This enables us to leverage the power of machine learning algorithms to analyze, classify, and extract insights from textual data.

3 Classification Task

As mentioned in the introduction, the objective is to train a model using the dataset that we get and test the model in the scrapped data. Firstly, we load the pre-processed tweet data and labels or targets of the complete dataset. To train it we decided to use a Super Vector Machine Classifier (SVM), with the training data to fit a good model to predict the labels in the text data. We choose SVM because is a popular choice for classification tasks because they used to provide high-accuracy results, they are robust to overfitting, can handle high-dimensional data and they provide a clear decision boundary to identify and interpret the results of the model.

Then, we test the model on the validation set extracted from the complete training dataset when using the split function. We left 20% of the complete dataset for validation. Finally, the accuracy is computed with the score of the predictions in the validation set. We get an accuracy value approximately equal to 0.73.

The discrepancy in performance among different models, where the Bag of Words (BoW) model provides the highest accuracy followed closely by the TF-IDF model, can be attributed to several factors:

1. Feature Representation: The BoW model and the TF-IDF model use different approaches for representing text data. The BoW model represents documents as a collection of word counts, disregarding the order of words, while the TF-IDF model considers the importance of words in the document. Depending on the specific dataset and task, the characteristics of the data may favor one representation over the other. For instance, if the dataset contains important keywords or relies heavily on word frequency, the BoW model might perform well.
2. Information Loss: The TF-IDF model incorporates additional information about the importance of words in a document, which the BoW model does not explicitly consider. This additional information can help the model better capture the distinguishing characteristics of the documents and improve accuracy. However, in some cases, this additional information may not be crucial or may introduce noise, causing the BoW model to perform better.

3. Dataset Characteristics: The performance of different models can vary depending on the specific characteristics of the dataset. Factors such as the size of the dataset, the distribution of classes, the presence of outliers or noise, and the complexity of the underlying relationships can influence the performance of different models. It is possible that the BoW model and the TF-IDF model are better suited to capture the inherent patterns and structure in your particular dataset, leading to higher accuracy compared to other models.
4. Model Complexity: The complexity and capacity of the models used (Keras and Doc2Vec) can also affect their performance. If the BoW and TF-IDF models have sufficient capacity and are well-optimized, they may outperform more complex models in certain cases. Additionally, simpler models like BoW and TF-IDF may be less prone to overfitting, especially when the dataset is small or when the other models are not properly regularized.

It's important to note that the performance of models can vary depending on the specific dataset and task. The accuracy rankings observed in your experiment may not hold true in general, and it is recommended to evaluate and compare models using multiple metrics and cross-validation techniques to obtain a more comprehensive understanding of their performance.

3.1 Evaluation of positivity and hate-speech through pre-trained transformers

In natural language processing, sentiment analysis plays a crucial role in understanding and extracting sentiments from textual data. Two popular approaches for sentiment analysis are rule-based transformers and embedded transformers. This section of the report aims to explore and compare the differences between these two approaches (which we have implemented in practice), with a focus on VADER as a rule-based transformer and pysentimiento as an embedded transformer based on BERT.

3.1.1 Rule-Based Transformers: VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based sentiment analysis tool widely used for social media text analysis. It employs a pre-defined lexicon of sentiment-related words along with a set of syntactic and grammatical rules to determine sentiment intensity. VADER assigns a polarity score to each token or sentence based on the presence of positive and negative words, negations, capitalizations, and other linguistic features. The final sentiment score is computed by combining these individual scores, providing an overall sentiment classification.

Here it is shown the evolution of the accuracy with respect to the threshold applied to Vader's compound score (the overall sentiment classification that we mentioned before).

The violin plot offers another insightful depiction of the data. It reveals that all political parties primarily maintain a neutral sentiment (reflected by a compound score of zero), with Partido Popular (PP), Podemos, Vox, and Ciudadanos all aligning in this range. However, PSOE's tweets exhibit a slightly positive sentiment.

Furthermore, the distributions of Partido Popular, Podemos, and Vox are concentrated around the center, indicating that the majority of their tweets have neutral sentiments, lacking strong positive or negative scores. On the other hand, PSOE and Ciudadanos display more dispersed distributions in terms of positivity. Their tweets are spread across a wider range, encompassing both neutral and positive compound scores. In contrast, the aforementioned parties' distributions exhibit denser clusters near the zero values.

Figure 2: VADER's accuracy wrt threshold

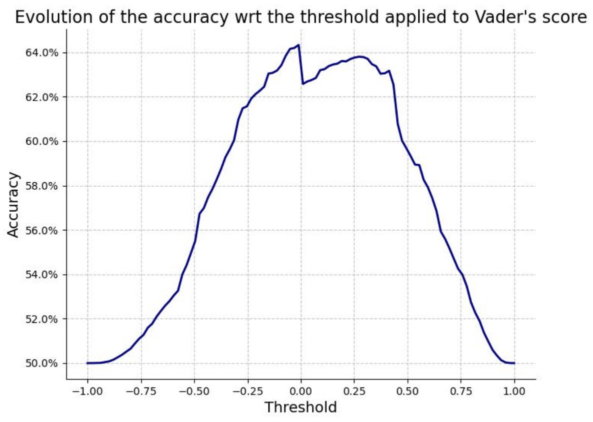


Figure 3: Violin plots

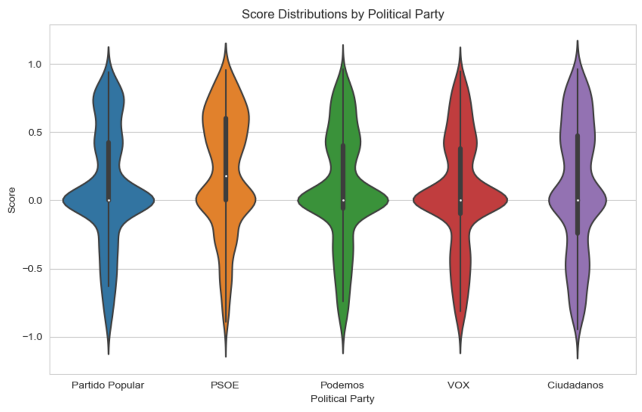
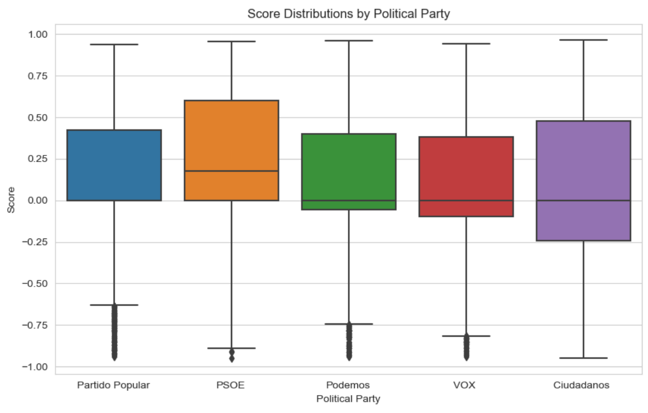
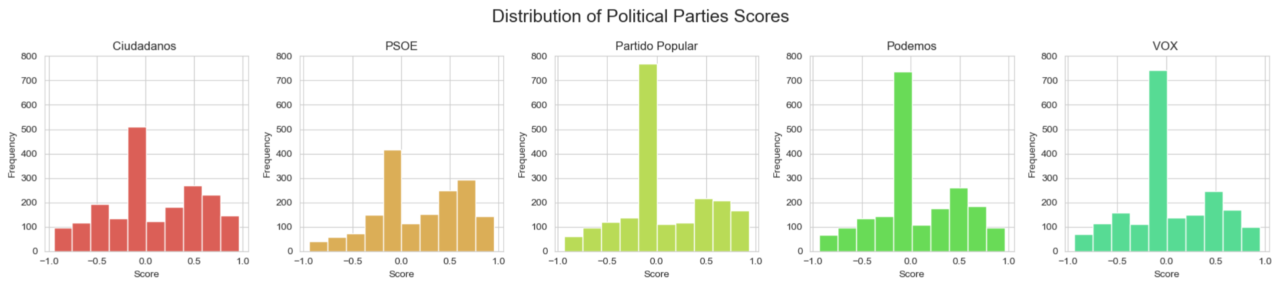


Figure 4: Political Parties' boxplots



When examining the boxplots, which provide clear insights into the distribution of quantiles, we find that the interquartile range (IQR) is relatively similar for Partido Popular, PSOE, and VOX. These parties demonstrate minimal inclination towards negative scores in their tweets. However, PSOE exhibits the widest IQR among all parties, indicating a greater range of sentiment scores. As mentioned earlier, their scores tend to be more positive compared to the other parties.
On the other hand, Ciudadanos displays a more balanced IQR compared to the other parties. This implies that the distances between the first and third quartiles in relation to the zero value are more evenly distributed.

Figure 5: Histograms per Political Force



Analyzing the histograms, we observe that Podemos and Vox exhibit similar score distributions. Both histograms resemble the one for Partido Popular in terms of positive and negative scores. However, Partido Popular's histogram shows a higher frequency of neutral scores in its tweets compared to the other two parties.
Regarding Ciudadanos (Cs), its distribution is slightly skewed towards the right, indicating a tendency towards more positive scores, although it remains relatively centered overall. On the other hand, PSOE's histogram displays a more pronounced right skew, indicating a similar frequency of positive scores and neutral scores. PSOE stands out as the party with the fewest scores close to neutral among all.

3.1.2 Embedded Transform bers: pysentimiento

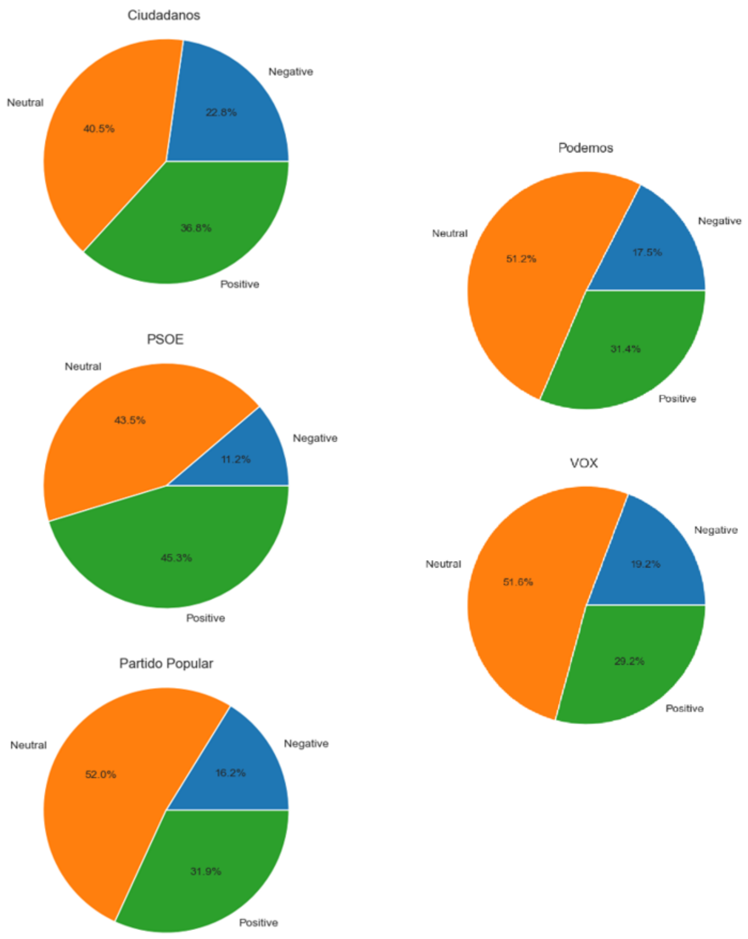
In recent years, embedded transformers have gained significant attention in sentiment analysis tasks. These models, such as pysentimiento, leverage deep learning architectures like BERT (Bidirectional Encoder Representations from Transformers) to capture complex patterns and relationships in text data. Unlike rule-based approaches, embedded transformers learn contextual representations of words and phrases by training on large-scale datasets. They generate word embeddings, which are dense vector representations of words in a high-dimensional space, capturing semantic relationships and contextual information. The sentiment analysis task is then performed by training a classifier on top of the embedded representations to predict sentiment labels.

Regarding some of our own insights, we have created pie charts (see Figure 6) with the frequencies of the tweets that score positive, neutral and negative per each party.

3.1.3 Differences and Advantages

The key differences between rule-based transformers like VADER and embedded transformers like pysentimiento are as follows:

Figure 6: Sentiment Analysis Pie Charts



- Flexibility and Adaptability. Rule-based transformers like VADER rely on handcrafted rules and dictionaries, which are designed for specific domains or languages. They may struggle to generalize well to different contexts or to adapt to new language patterns. In contrast, embedded transformers like pysentimiento have the advantage of learning from large-scale data, allowing them to capture context-specific nuances and adapt to various domains and languages.

- Performance and Accuracy. Embedded transformers often outperform rule-based transformers in terms of accuracy and performance. The deep learning architectures in embedded transformers, such as BERT, can capture complex language patterns and understand the context in a more sophisticated manner. This enables them to handle a wider range of sentiments, sarcasm, and subtle feelings, leading to more accurate sentiment predictions.

- Interpretability. Rule-based transformers like VADER offer interpretability as they provide explicit rules and dictionaries for sentiment analysis. This allows users to understand the reasons behind the sentiment predictions. On the other hand, embedded transformers like pysentimiento, while achieving higher accuracy, may be less interpretable due to the black-box nature of deep learning models. Understanding the internal workings and making sense of the decisions made by these models can be challenging.

4 Implementation of the dashboard

Dash implementation provides a seamless platform for developing interactive web applications with customized data visualization using your own Dash plots. With Dash, you have the flexibility to create dynamic and interactive plots tailored to your specific needs.

In our particular case, we have opted for focussing on sentiment analysis, interactive wordclouds per political force and the LDA (applied for the Kaggle dataset). The following figures are examples of our dashboard:

Figure 7: Range of scores by sentiment

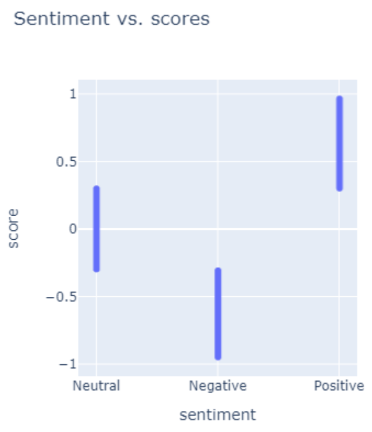


Figure 8: Distribution of sentiment labels (0: Negative, 1: Positive) along the dataset

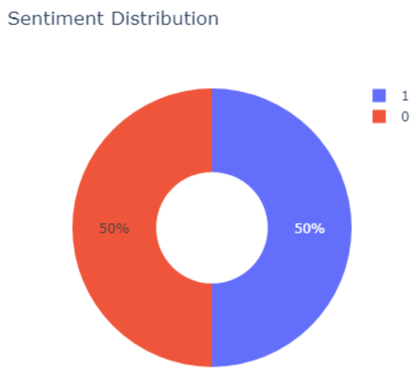


Figure 9: Sentiment Analysis boxplot by political force

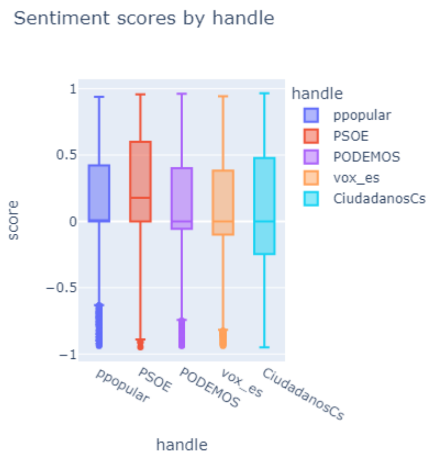


Figure 10: LDA topics

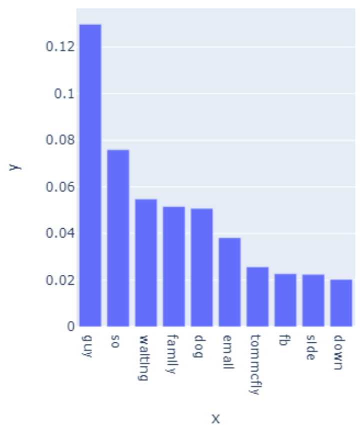
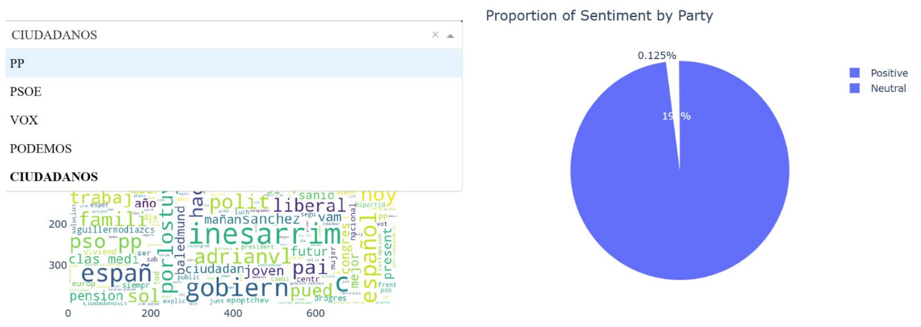


Figure 11: Interactive wordcloud and General Sentiment per Party



5 Acknowledgement of authorship

The provided code has been developed as an essential component of our Machine Learning applications project, drawing inspiration from the work and ideas contributed by various authors and sources. We extend our sincere appreciation and gratitude to these individuals for their valuable contributions, which have significantly influenced the development and success of our project.

1. The explanation of Twitter's scrapping methods with Python were extracted from:
   - Data Flair. How to scrape data from Twitter using Python
   - FreeCodeCamp. Web Scraping with Python – How to Scrape Data from Twitter using Tweepy and Snscrape
   - Text mining - How to pull data from Twitter in Python
2. The source code of pysentimiento's analyzers were extracted from the following documentation.
3. The theoretical concepts and implementation for NLP pre-trained transformers were extracted, adapted and summarized from the following sources:
   - Towards Data Science. Understanding Political Twitter
   - Neptune AI. Sentiment Analysis in Python: Textblob vs Vader vs Flair,
   - Towards Data Science. Basic Binary Sentiment Analysis using NLTK
   - Towards Data Science. Step by Step Twitter Sentiment Analysis in Python
   - NLP Sentiment Polarity Classification with Kaggle
   - AnalyticsVidyha. Twitter Sentiment Analysis Using Python — Introduction & Techniques
   - Sentiment Analysis: VADER or TextBlob?
4. Notions of topic modelling and visualization with gensim were extracted from here:
   - Machine Learning Plus. Topic Modeling with Gensim (Python)
   - Topic modeling visualization – How to present the results of LDA models?
5. The pipeline to translate our database from Spanish to English was adapted from here: Towards Data Science. Translate a Pandas data frame using googletrans library

Please note that the code has been modified and adapted for the specific context of our project and the datasets that we were using.