Presidential preferences in Colombia through Sentiment Analysis

Edwin Puertas
Dept. Engineering of Systems
Universidad Tecnológica de Bolívar
Cartagena, Colombia
epuerta@utb.edu.co

Juan Carlos Martinez-Santos
Dept. Engineering of Systems
Universidad Tecnológica de Bolívar
Cartagena, Colombia
jcmartinezs@utb.edu.co

Pablo Andrés Pertuz-Duran
Dept. Engineering of Systems
Universidad Tecnológica de Bolívar
Cartagena, Colombia
ppertuz@utb.edu.co

***Abstract*—Abstract.**
***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In May 2022, a presidential debate occurred between the candidates running for the presidency of the Republic of Colombia. In said debate, people could give their opinions about the proposals presented and the answers submitted by each applicant in said debate. Therefore, sentiment analysis was carried out during the Colombian 2022 presidential election debate, resulting in who are those candidates who generated more positive feelings or affinity towards the candidate and negative ones who did not share the candidate's ideas. This paper presents the methodology we follow to capture and process the data and the steps to train the classifier. The results show space for improvement in the path to identifying feelings from comments gathered from social media.

II. Related work

To carry out this project, we took some reference articles into account, which serve as evidence of how the neural network with which we work in this project works and thus corrects errors or implements improvements.

Among the master's thesis, the one by Alva Segura and Daniel Abraham stands out. In their paper, they implemented a sentiment analysis in the congressional elections of the year 2020 in Peru with the technological tools in the market that solve sentiment analysis problems [1].

Another that stood out was Mauro Callejas and Manuel Vélez. Their article proposed a hybrid model that predicted the outcome of the first round of presidential elections in Colombia in 2018, aiming to improve the quality of the final prediction and minimize errors. Here, the activities of Facebook and Twitter users were recorded and analyzed by artificial intelligence algorithms, obtaining accurate and consistent results [2].

III. Methodology

A. Data Sets

For the training of the models, a set of data was created based on the opinions of Colombians on the social network Twitter, referring to the Presidential Debate that took place on May 23, 2022. we initially extracted these tweets through the Tweepy library that allows access to the Twitter API.

We asked the company to access the Twitter API for an account with developer privileges. In the Twitter developer portal, we created a new application with four attributes necessary for authentication: consumer token, consumer token secret, access token, and access token secret. Then, we create an OAuthHandler instance that receives the previously obtained consumer token and consumer token secret as arguments. The *set_access_token* attribute of the object created for the authenticator has the access token, and the access token secret is also an argument. Once the OAuthHandler gets an access token, we initialize the API method of the library to authenticate us on Twitter and perform the extraction tasks.

The objective was to extract 2,000 tweets with the hashtag #ElDebateDefinitivo, published in Spanish as of May 24, 2022. This request was made with the Cursor method of tweepy, selecting 2000 tweets that met the conditions of the request. Then, we store the tweets in a dictionary, saving the information with the label 'Tweet' and the date of publication of the comment with the key 'Timestamp'. After this, three lists were defined to manage the multiple aliases that the selected candidates could receive in the tweets, being a list for the aliases of the candidate Gustavo Petro ('GustavoPetro', 'Gustavo Petro', 'Gustavo', 'Petro' , 'petrogustavo'), a list for candidate Federico Gutierrez ('FICO', 'Fico Gutierrez', 'Fico', 'Federico', 'Federico Gutierrez', 'FicoGutierrez') and a list for candidate Rodolfo Hernández (' RodolfoHernandez', 'Rodolfo Hernandez', 'Rodolfo', 'Ingeniero Rodolfo', 'ingrodolfohdez'). Finally, we chose these three candidates because they were the most likely to win the elections, according to almost all the polls carried out on Twitter.

Once the three lists were established, the function *identify_subject* was defined. This function iterated the dictionary with the stored tweets, identifying which candidate or candidates appeared in the comment. With this information, we created a DataFrame with five characteristics/columns: Tweet, Timestamp, Petro, Fico, and Rodolfo. The first two columns correspond to each tweet's text information and publication date, respectively. In contrast, the Petro, Fico, and Rodolfo columns had values of 0 or 1, indicating whether the tweet mentioned the candidate Gustavo Petro, Federico Gutierrez,

or Rodolfo Hernández, respectively.

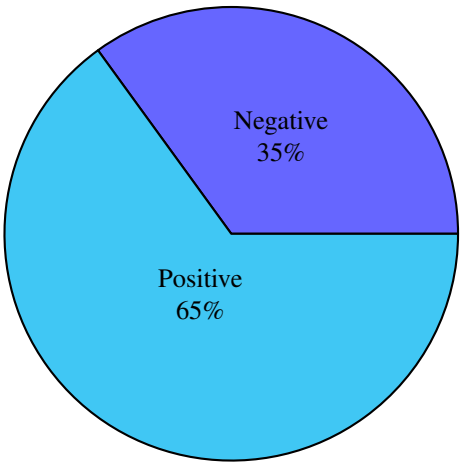
We remove duplicate comments, keeping the first to appear, going from having a DataFrame with 2000 tweets to one with 246 tweets.

From this DataFrame, we created three more data frames, separating the tweets that mentioned only one candidate to avoid ambiguities in training. We made the first DataFrame corresponding to the comments that said only the candidate Gustavo Petro (79 tweets). In the same way, it was done with the second and third DataFrame, storing the words corresponding only to Federico Gutierrez (37 tweets) and Rodolfo Hernandez (10 tweets), respectively. Each DataFrame had two features/columns, the text comment 'Tweet' and the corresponding 'label', which we set to an empty/null value NaN at this stage of the process. Finally, we exported the three DataFrames created to CSV format using the Pandas library.

DataFrames were labeled by 3 college students, placing a value of 0 in the 'Label' column when the comment represented a negative sentiment, 1 when it represented a positive sentiment, and leaving the box empty/NaN when the comment represented a neutral sentiment.

Then, the 3 labeled DataFrames were imported and concatenated into one, ignoring the indexes. The new DataFrame had 150 entries, of which 123 had non-empty cells in the 'label' column. The entries with empty values (neutral comments) were dropped, resulting in a DataFrame with a total of 123 entries. This final DataFrame was exported in CSV format with utf-8 encoding and represents the data set used for model training.

The data set used for training has 123 entries or comments (0-122), of which 80 represent positive sentiment and 43 negative sentiment. There are no null values.



B. Processing

After obtaining the data set, we applied text mining techniques to normalize the content of the messages, remove punctuation marks and special characters, and elements that are not of interest to the study, such as hashtags, emojis, and mentions and links, among others. Based on the above, we used the TextAnalysis library to facilitate tasks related to text [3]. Algorithm 1 described the functionalities of the library. Terms with a frequency per document of less than two were also limited, which reduces the number of non-informative characteristics. Finally, we applied a stratified random sampling technique to avoid bias in the test to guarantee the balance between classes.

Algorithm 1 Preprocessing [3]

Input: Set of messages which can be denoted as:

$$M = \{message_1, message_2 ... message_n\}$$

Sentence: Set of words that express judgment with full meaning and syntactic autonomy.
Tasks:

- Remove users: Used to remove mentions of users in comments.
- Remove hashtags: Used to remove hashtags in comments.
- Special Patterns. This task will remove special characters such as @, *, #, $, etc.
- Remove av: Used to remove tagging images or videos in comments.
- Remove emoji: Used to remove emojis in comments, based on the clean text library
- Normalize. This task involves executing all the previous tasks,

Output: Dictionary with the result of all the tasks.

C. Feature Extraction

*1) **Bidirectional LSMT**:* To convert the words of the tweets to vectors, we applied the TextVectorization layer from the TensorFlow library. TextVectorization is a preprocessing layer that maps text features to entire streams.

To perform the vectorization process, a vocabulary must be supplied to the TextVectorization layer. This was learned by the layer with the adapt() method, which analyzed the preprocessed training data set, calculating the frequency of occurrence of each individual text string (words), and then creating a vocubalary for them. The vocabulary size was limited to 200000 features or words and in case the entry has more than 200000 unique words, the 200000 most frequent words will be used to create the vocabulary. The output sequence was also defined to have a size of 1800 elements.

The vocabulary is composed of the most frequent words and an index to reference them. Table 1 shows some of the words in the vocabulary and their respective index.

For instance:

"Seguridad, salud y empleo es lo que le falta al pais."

A one-dimensional tensor of shape 1800 is generated, with the array of elements: [58, 30, 7, 124, 13, 17, 4, 20, 367, 38, 46, ..., 0, 0, 0], where each element of the array represents the index of each word of the original sentence in the vocabulary. As the input comment does not exceed 1800 words, the rest of the array elements are filled with 0 until a shape of 1800 is completed (Zero-Padding).

TABLE I
VOCABULARY EXAMPLES

Word	Index
debate	16
propuestas	27
dice	29
salud	30
voto	33
colombianos	36
candidato	37
presidente	45
pais	46
sistema	57
seguridad	58
gobierno	66
cambio	74
votar	75
social	83

Finally, we applied vectorization to each of the comments in the training and validation data set, making these datasets suitable for the model.

2) ***BERT based model****:* For feature extraction, we created two functions: Convert data to examples and Convert examples to TF Dataset.

Convert data to examples: This function has the training and validation data frames, and the names of the columns of the independent variable (the tweets) and the target or objective variable (labels) as parameters. It uses the transformers library's InputExample function to create examples, a format supported by transformers models that consist of two pieces of text (text_a is the tweet and text_b is empty) and the corresponding tag.

For instance, a preprocessed tweet:

"seguridad salud y empleo es lo que le falta al pais"

Generates:

InputExample(text_a='seguridad salud y empleo es lo que le falta al pais', text_b=None, label=0)

This process is applied to each comment in the training and validation data, saving the generated examples.

Convert examples to TF Dataset: This function has the output generated by the convert_data_to_examples process, the initialized tokenizer (for this job, we use a tokenizer based on the bertbaseuncased model), and a truncate of a maximum of 128 tokens (default) as model input. It tokenizes the generated examples and prepares them to be suitable inputs to the model, using the encode_plus method:

- Add special tokens.
- Sets a maximum length.
- Returns the IDs of the tokens.
- Returns attention masks.
- Padding.
- Truncation.

Finally, it generates a Dataset using tensorflow's tf.data.Dataset.from_generator() method, storing the input_ids, attention_mask, token_type_ids and the label of each example.

Input_ids are the indices of each word in the comment with respect to the vocabulary downloaded when importing the BERT-based model. Attention_masks is an array with elements of value 0 or 1, indicating whether the attention mechanism of the BERT model should be applied to a token or not (it will have a value of 1 if the token corresponds to a word in the comment and 0 if the token corresponds to zero-padding). Token_type_ids is set as a zero value array by default, and the label corresponds to the label for the comment.

The function is used to generate a Dataset for the training data and a dataset for the validation data.

D. Training

For training, we iterated the training and validation data distribution concerning the base dataset, between 80-20, 75-25, and 70-30. The learning rate value was iterated between 1e-5, 2e-5, 3e-5, 4e-5 and 5e-5. In addition, we used the following metrics Accuracy, Precision, Recall, and F1. Accuracy: Measures the percentage of cases in which the model was correct. Precision: the quality of the model when classifying (number of correctly classified positives over the total number of classified positives). Recall (Completeness): Number of correctly classified positives over the total number of real positives. F1 Score: This value combines the precision and recall measures into a single value. It is practical because it makes comparing the combined accuracy and recall performance between various solutions easier.

We used the Keras EarlyStopping class to monitor the training of the models. It uses seven parameters:

1) The metric to monitor.
2) The minimum increment or decrement that the metric must undergo in order not to stop training.
3) The number of epochs to wait before stopping training.
4) Whether or not to show details during training (verbose).
5) Set whether to search for a decrease or increase in the metric to monitor.
6) If you want to compare the performance of a model with a baseline.
7) If you want to restore the model values (checkpoint) at the time that presented the best performance.

IV. EXPERIMENT

To evaluate the best training and validation distribution, we used a baseline with a learning rate of 1e-5 for the optimizer, loss function binary_crossentropy, and binary_accuracy as metrics for the Bidirectional model, and sparse_categorical_crossentropy of loss function or sparse_categorical_accuracy as for the BERT model.

Five iterations were performed where the loss function for the Bi-LSTM model was compared with a learning rate of 1e-5, 2e-5, 3e-5, 4e-5, and 5e-5 and a distribution of 70%-30%. Between the training and validation data.

A. Best Model

Table III indicates the best value for accuracy and f1 score obtained by each model during the iterations, followed by the

TABLE II
EXPERIMENT MODEL SCORES

Model	Precision	Recall	F1	Accuracy	Learning rate
Bi-LSTM	0.72	1	0.83	0.69	1e-5
BERT	0.83	1	0.85	0.76	1e-5
Bi-LSTM	0.66	1	0.80	0.66	2e-5
BERT	1	1	0.818	0.76	2e-5
Bi-LSTM	0.66	1	0.80	0.66	3e-5
BERT	0.81	1	0.81	0.73	3e-5
Bi-LSTM	0.66	1	0.80	0.66	4e-5
BERT	1	0.95	0.79	0.66	4e-5
Bi-LSTM	0.66	1	0.80	0.66	5e-5
BERT	1	0.95	0.79	0.73	5e-5

training and validation data distribution and the learning rate corresponding to the iteration.

TABLE III
BEST MODEL SCORES (70%-30%, A=1E-5)

Model	Accuracy	F1 score
Bi-LSTM	0.69	0.84
BERT	0.76	0.85

To choose the best model, we will consider 2 assumptions [4], [5]:

- Accuracy works best when true positives and true negatives are most important, while the f1 score works best when false positives and false negatives are crucial.
- Accuracy works better when the class distribution is similar, while the f1 score is better when the classes are unbalanced.

For our application, false positives and negatives are crucial, and the classes are unbalanced, so we will choose the model that presented the best f1 score, the BERT model trained with a learning rate of 1e-5.

V. CONCLUSION

We evaluated the best-chosen model using six comments. He classified the sentiment of the comments correctly in all cases, but his decision was not so sure in one of the comments (score = 0.5991), showing that he has a lot of room for improvement.

ACKNOWLEDGMENT

The preferred spelling

REFERENCES

[1] D. A. Alva-Segura, "Análisis del sentimiento político en twitter durante las elecciones congresales 2020 en el perú," Master's thesis, 2021.

[2] M. C. Cuervo and M. A. V. Guerrero, "Predicción electoral usando un modelo híbrido basado en análisis sentimental: Elecciones presidenciales de colombia." *Revista Politécnica*, vol. 15, no. 30, pp. 94–104, 2019.

[3] E. Puertas, L. G. Moreno-Sandoval, J. Redondo, J. A. Alvarado-Valencia, and A. Pomares-Quimbaya, "Detection of sociolinguistic features in digital social networks for the detection of communities," *Cognitive Computation*, vol. 13, no. 2, pp. 518–537, 2021.

[4] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[5] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, "Finding the best classification threshold in imbalanced classification," *Big Data Research*, vol. 5, pp. 2–8, 2016.