**Explanatory Document Predictvia AI Challenge 2020**

Team: CMD

members:

-Mario Catapano

-Massimo Di Gennaro

**1. Pre-processing:**

**-Methods used:**

● Group web page links: all URL addresses were replaced by a common word (direccionweburl), to group the different web links.

● Group all numbers: Any numerical figure that appeared in the tweets was replaced by a common word (valornumerico)

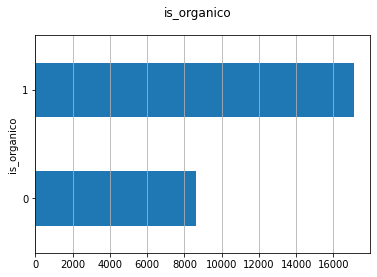
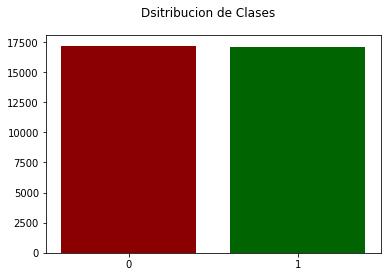
● Group all Twitter users: all users kept in tweets (@userexample) were replaced by a common word (twitteruser)

● Separate letters and special characters: all the words were separated from the possible special characters (,.?:'"() etc.) that they had attached, for example a tweet like "Today, or tomorrow?" would become “Today , or tomorrow ?”. This was done so that they are not considered by the model as different words (this step could not be done before since URL addresses have symbols that, if they are separated from the letters, the address stops working or is not recognized by the algorithm).

● Hashtags in the dataset: hashtags (#) are not grouped as it is done with the (@) and URL symbols, because these sometimes have a word that gives context to the phrase, such as #music or #architecture (or words that are not in Spanish but recurring such as COVID19). However, to extract the word, the '#' is separated from the text and the '#' symbol is replaced by a '##', since there are occasions of false tweets where the # is separated from the text, and if we do not make this change there would be no difference after preprocessing a "#spain" for example from a “# spain”, with this way of preprocessing the difference would be evident since these hashtags would result in "## spain' and "# spain" respectively.

● Upsampling/Oversampling: by applying this strategy, which consisted of doubling the tweets of the minority class in the data (inorganic tweets), we managed to raise the F1 score by approximately 0.005.

Before Oversampling: After Oversampling:

● Feature Engineering: Like the Upsampling technique, this technique improved our F1 by approximately 0.005. Among the data we extract from the tweets for Feature Engineering are: number of words, number of characters, number of capitalized words, number of words that begin with a capital letter, etc. And using the Spacy library, we extracted data such as: number of verbs, number of stopwords, number of digits/numeric figures, number of adjectives, number of organizations, number of unknown and known words, number of punctuation marks, etc. In total, 34 different Features are extracted.

● Tokenization and Padding: finally, the tweets were tokenized (at the word level) and zero padding was used on the right, the maximum tweet length was 115 words (the real tweet with the most words after preprocessing had that length)

**-Discarded methods:**

Below, we mention the preprocessing methods that we decided not to implement after several tests.

● Lowercase: It was experimented using the same model (Embedding and LSTM) and the same validation data, with the only difference being that in the preprocessing of one attempt all the letters were changed to lowercase and in another they were not, and the following results were obtained:

|  |  |  |
| --- | --- | --- |
|  | Keeping Capital Letters | Discarding Capital Letters |
| LOSS | 0.1292 | 0.1335 |
| ACC | 0.9522 | 0.9481 |
| F1 | 0.9506 | 0.9457 |

It can be seen that the results obtained with the use of capital letters are superior and, therefore, the capital letters were left in the tweets.

● Language analysis using Langdetect: it was tested by removing all the tweets that were not in Spanish, however, many tweets that were not really in another language were lost, therefore, then it was experimented with removing only those that were in English, but it was not used in the end since it decreases performance and the validation set had other languages ​​too (some tweets in English mainly).

● Lemmatization: it was decided not to use this method, since some of the words were not correctly converted to their root, as, for example, in the case of the word “serie”, the lemmatization converted this word to “seriar”, which is not correct.

● Remove stopwords: This pre-processing method was discarded as it was lost too many words that we considered important for the model, since stopwords in Spanish are many more words than stopwords in English.

● Remove punctuation: in the case of the score, this method was discarded since we consider that the score can provide important clues to the model, when predicting whether a tweet is organic or inorganic, for example, the number of signs of punctuation can be an important factor within Feature Engineering, and for this reason, it was decided not to remove it.

● Group emojis and hashtags: after several experiments, we decided to discard this option as it does not improve the results, although it does increase the number of non-vector words in the pre-trained embedding later on.

**2. Model tests:**

Several tests were performed on the model, to find the optimal hyperparameters of the LSTM network, these tests were saved in a Google document and later analyzed.

Regularization was not used, since the best results were obtained without using Dropout.

For the validation set, 30%, 20%, 10%, 5%, 1% were used, and the best results were obtained with 20%.

Different embeddings were tested, specifically, not pre-trained 50, 100 and 300 dimensions, and pre-trained of 100 dimensions and 300 dimensions, however, for the final project the 300-dimensional embedding was used, as it provided the best results.

**3. Code and reasons for its use:**

To read the data, we use a try-except, so that, for those who do not have access to our Google Drive, the data is loaded directly, either from its address in Kaggle (../input/aichallenge2020/training.csv) or directly (training.csv).

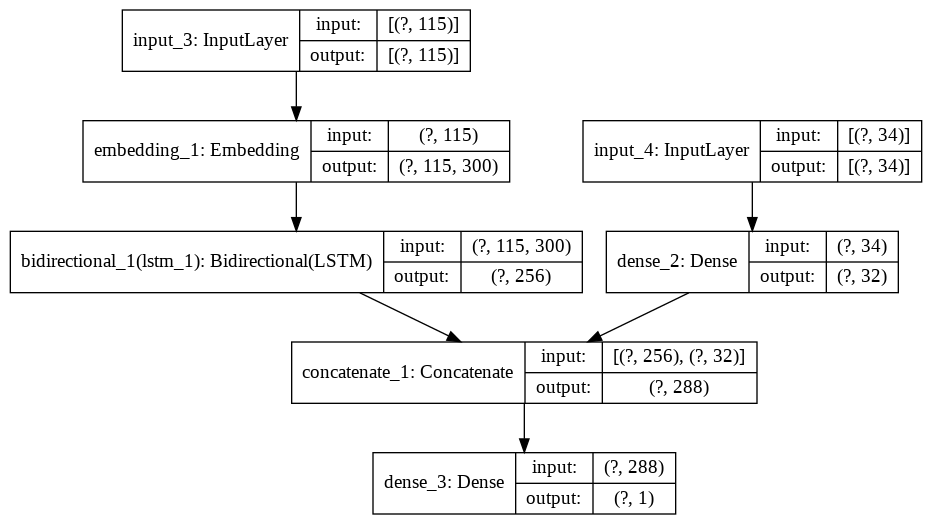
The number of organic and inorganic tweets in the dataset was analyzed, and finding that there is an imbalance, it was decided to use Upsampling to balance the number of inorganic tweets (will be explained later). The most and least used words in the dataset are searched for as an exploratory analysis of the data, and also to find those that could be counted in Feature Engineering, and that appear a lot in one class, but not in the other, and these were: asterisks and hashtag (they appear a lot in inorganic tweets) and hyphens (they appear a lot in organic tweets). The longest real and fake tweets, both in the training set and in the validation set, were then analyzed as a basis for determining the length to select as the maximum length of tweets (115 was selected at the end).

The most and least used words are then revisited, and the changes are noted. For padding and tokenization, a total of approximately 51,000 different words were used, when counting the different words in the dataset. In turn, a maximum length of 115 was selected, after verifying that this is the number of words in the longest organic tweet.

The correct operation of the Tokenization is checked, by converting the tokens back into text and verifying that it is the same text of the original preprocessed tweet. Then, a pre-trained embedding, downloaded from zenodo.org (https://zenodo.org/record/3234051/files/embeddings-l-model.vec?download=1), is used, and the vectors of 300 dimensions are read and obtained of each word within the embedding that coincides with the words that remain in the dataset after preprocessing (in this case if we make the comparison with the words in lowercase, in such a way that the same vector is assigned to Casa or casa, for example), with approximately 44,000 words found and 6,000 without embedding vectors (for words without embedding, their vector is initialized using a uniform distribution between -0.5 and 0.5).

The minority class is upsampled, in this case, the inorganic tweets, so that both categories have a very similar amount of data, and then the amount of organic and inorganic tweets is verified again.

Then, the training and validation data are separated, the functions for the F1 score are calculated, and the model is created with the following characteristics:



The model has two inputs, one with the training sequences (tweets), and the other with the 34 feature engineering columns, the first is passed through the Embedding layer, which despite being pre-trained, we still train it because we discovered that this is how we get better results, and then by a 128-neuron bidirectional LSTM, while the feature engineering is passed by a 32-neuron Dense layer, Relu activation. These two layers are then concatenated, and passed to a final Dense layer of 1 neuron and sigmoid activation to obtain the probability that the tweet is organic or inorganic. Training: Adam optimizer, batch size of 32, 2 epochs, and with binary-crossentropy loss function.

Finally, the model is saved, as well as its representative image (shown above), and predictions are made using this model on the test set. These predictions are then saved to a CSV file to be uploaded to Kaggle.

**4. Other Models and Architectures tested:**

**4.1. Deep Learning:**

● Bidirectional LSTM:

This model had the embedding layer, followed by a bidirectional LSTM layer (with which 64, 128 and 256 neurons were experimented with) and the last dense (1) sigmoid layer. With this model, the highest F1 score obtained was approximately 0.9

● LSTM with convolutions:

Same as the previous model, with the difference that, after the embedding layer, there were a couple of extra layers with CONV1D and max pooling, the results obtained were slightly below those obtained by the model without convolutions.

●Transformers:

This model had the Encoder structure of the Transformers architecture, with a single Transformers block of 32 heads, the highest F1 score obtained with this model did not reach 0.9.

● Convolutions:

A model that after the Embedding layer was made up of several CONV1D and max pooling layers, which would later become Fully Connected layers. The results were quite similar to the LSTM-only model, and we ended up discarding this one for the LSTM one.

**4.2. Machine Learning**

It was decided to carry out tests using Machine Learning models, the results of which are shown below:

|  |  |
| --- | --- |
| Method | Accuracy |
| Logistic Regression | 0,66 |
| Support Vector Machine (SVM) | 0,67 |
| Random Forest (101 nodes) | 0,78 |

As can be seen, the results obtained are below the results obtained through the use of Deep Learning (LSTM), so these were discarded.