Text classification - YouTube viral comments

Mario García García

May 11, 2023

Abstract

This project presents a beta-version classification model for YouTube comments to identify those that are most likely to go viral. To achieve this, a combination of sentiment analysis and machine learning techniques were used to identify relevant patterns and features in the text. Data from previously viralised comments from "La Velada del Año 2" of the famous Youtuber Ibai Llanos were collected and labelled and used to train and validate the proposed model. This approach can be useful for advertising and marketing companies looking to identify and leverage viral comments to promote products or services online before humans tell a certain comment whether they liked it or not.

1 Introduction

This project aims to leverage the large volume of comments available on YouTube to develop a ranking model that allows content creators and companies to identify the comments with the highest potential to go viral. This can be of great help for the promotion of products and services, as viral comments have the potential to influence user perception and increase the visibility of a brand. In this case, we have taken into account all the comments from the famous YouTuber and streamer Ibai Llanos' Year 2 Evening event, with the aim that when Year 3 Evening takes place, we will be able to get viral comments.

The model development process involves several stages, from data collection and cleaning, data analysis to implementation and evaluation of the final model. Text analysis and data cleaning are important to ensure that the model is working with high quality data and to reduce noise in the data that could affect the accuracy of the model.

Sentiment analysis using BERT [6] and nltk [3] transformers is a powerful technique for understanding the tone and sentiment of comments. Combining this technique with machine learning and recurrent neural networks allows to identify patterns in the data and perform text classification, which allows to identify the comments that have the highest potential to go viral.

In short, the development of a YouTube comment classification model to identify which comments are most likely to go viral is an exciting and challenging project that combines text analytics, machine learning and neural network techniques. If successful, this model could have a major impact on online marketing and advertising, as well as on users' perceptions of products and services, and not just for this particular case.

2 Project explanation

2.1 Data collection

In order to develop a YouTube comment ranking model to identify YouTube comments that are most likely to go viral, relevant and representative data is needed. In this project, a JavaScript code has been created using Google Sheets for the collection of YouTube comment data. The data has been obtained from all comments of "La Velada del Año 2".

From the data collected, only two columns have been selected: the content of the comment and the number of likes that a comment has received. These columns were selected because they are the most relevant and useful data for identifying comments that are most likely to go viral. In addition, the feature representing the name of the user who left the comment has not been used in order to anonymise YouTube users.

Comments that reply to the first comment have also been removed, as it is felt that they may be influenced by the initial comment and may not be as likely to go viral on their own. In addition, the entire comment thread has been removed, because YouTube users prefer to read only the original comment rather than go through an entire comment and reply thread.

The approach of selecting only the most relevant data and drop comments that may be influenced or biased allows a more representative sample of YouTube comments to be obtained for training the comment classification model. In this way, it is expected that the model will be able to identify patterns and features relevant to viral comment prediction with greater accuracy and generality.

The code for this script can be obtained directly from my GitHub.

2.2 Data analysis

There are not clear definition of what is a viral comment and there is no clear threshold point to determine to what extent a comment has gone viral or not. However, in this project, it has been established that any comment that receives more than 20 likes is considered a viral comment. This criterion has been adopted for practical predictive purposes and will allow comments to be classified into two categories: viral and non-viral. In this way, YouTube's comment classification model will be able to identify relevant patterns and characteristics that can help predict viral comments more accurately.

In the same way, it is worth showing how Latin American influence outweighs Spanish influence:

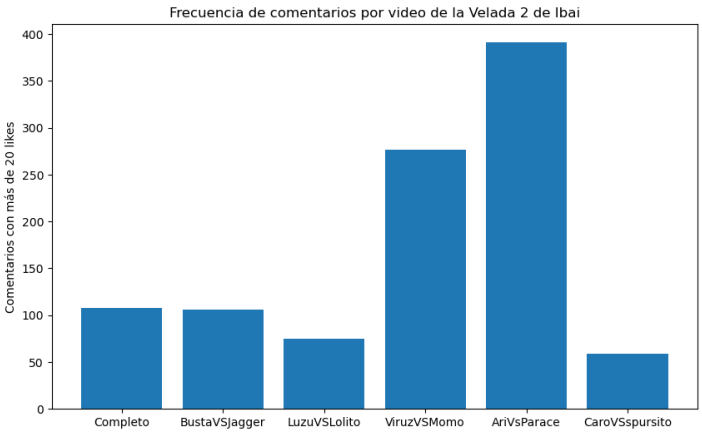


Figure 1: *On the x-axis we represent for each video the frequency of comments with more than 20 likes along with the fight that took place. The name "Complete" is the complete raw video.*

This can be explained by the fact that in Latin America there are an estimated 237 million people in the 10-24 age group, compared to Spain, where the total number of people in any age group is 47 million, almost 6 times more.

After append all the comments in the same database, two types of counts have been made; the most significant one is to see which words are the most repeated in all the videos and the number of prepositions, articles, conjunctions...etc.

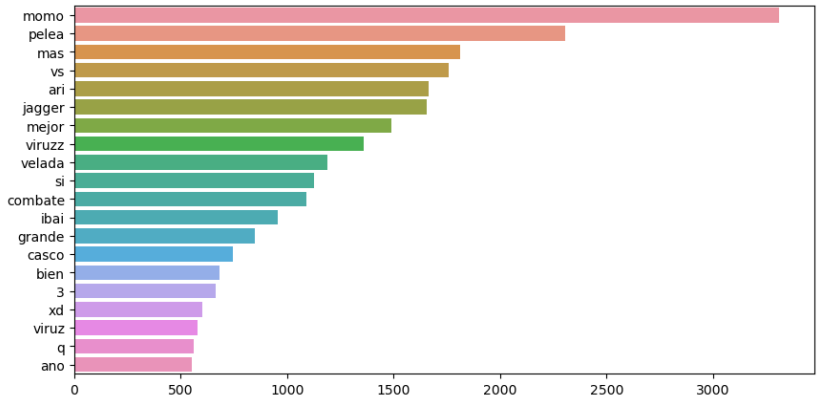


Figure 2: *Frequency of most used words in "La Velada del Año 2"*

You can also see here the support and virality that Momo and Arigameplays have involved in this evening. More interesting data is the impressive repercussion that byViruZz has had, in second place with momo, since viruz and viruzz are spelled in a similar way. Finally, it is worth mentioning as a key word "helmet" or "casco", perhaps because of the concern that each boxer had a different one.

Here is shown all the most common prepositions, articles, conjunctions...etc. in these comments that have been removed.

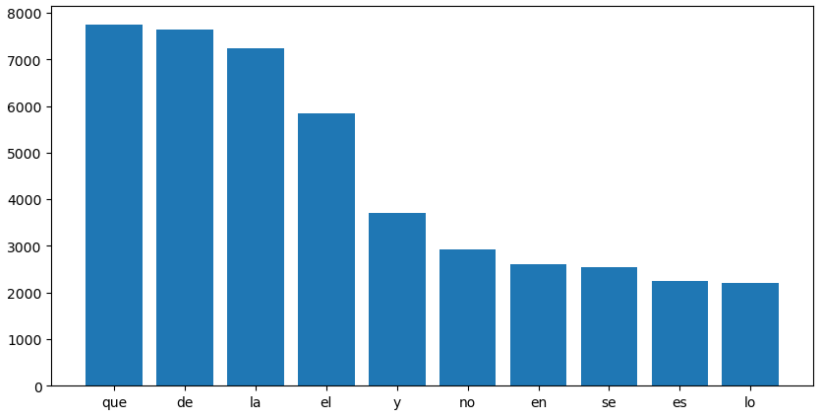


Figure 3: *Frequency of most used stopwords of "la Velada del Año 2"*

2.3 Data pre-processing

The possibility that an original comment with replies is more viral than those without replies has also been explored, as it could provide valuable information for our model. However, it has been determined that the presence of replies in a comment is not necessarily an indicator of its virality. Therefore, this idea has not been included in the prediction model.

In natural language processing, it is common to come across texts that contain elements that are not relevant to the analysis we want to perform. For example, special characters and punctuation, such as exclamation marks or commas, may be useful for the meaning of a text, but not for our analysis objective. For this reason, one of the most popular cleaning techniques is to remove these elements.

Also, numbers are usually not relevant in natural language analysis, unless you are doing a specific analysis on them. Therefore, removing numbers from the text can facilitate data processing.

Another very useful cleaning technique is the elimination of empty words or stopwords, which are very common words in a language that do not contribute relevant meaning to the analysis. These words are usually articles, prepositions, conjunctions, among others. By removing them from the text, we can reduce the noise and obtain a clearer and more concise version of the text.

Nowadays, emojis have become very popular in written communication, especially on social media. However, in many cases they are not relevant to the analysis we want to perform. For this reason, one cleaning technique is to remove emojis from the text.

It is common to convert all text to lower case to prevent natural language processing from considering two identical words as different just because they are written in different upper or lower case. In this way, we can ensure that our analysis is more accurate and consistent.

Some specific characters that do not provide relevant information for the analysis have also been removed. In particular, characters such as "", which are HTML tags used to close bold YouTube comments or to reference certain moments in the YouTube video, have been removed. Comments containing references to web pages have also been removed through the use of "http" and "</br>" tags, which are used for line breaks.

These specific characters have been removed because they do not provide relevant information for the identification of patterns or characteristics in viral comments. In addition, the presence of these characters can hinder data analysis and subsequent interpretation. Therefore, their removal helps to simplify the data and improve the quality of the results obtained.

Finally, it is important to mention that in the use of language we can often encounter words with accented spellings, such as "leon" or "león". Although our brain can recognise the word correctly in both cases, it is good practice to remove the accents in order to maintain consistency and uniformity in comments. This avoids confusion and facilitates natural language processing for the task of predicting viral comments on YouTube.

With the changes in the database structure, a sentiment analysis has been done using a deep learning transformer model called BERT [6] [2], which has allowed the creation of a new column that provides valuable information about the sentiment of the comments. It should be noted that BERT is a very complex model that requires a large amount of time and resources to process all the comments in the database. Because the variable to be predicted (virality) presents an imbalance in the data, it was decided to reduce the database to 8700 records, keeping the same number of viral comments and eliminating 20,000 records of non-viral comments. This has significantly accelerated the sentiment analysis process, reducing the run time from several days to 15 hours. This decision was taken in order to obtain faster results and for a first approximation of the model.

Another sentiment feature has also been created, this time a simpler one, using NLTK [3]. I am going to have during training comments and sentiment information.

One of the common problems that can be encountered when working with machine learning models is class imbalance in the data. This means that there is a minority class that has too few examples compared to the majority class, which can negatively affect the performance of the model and its ability to correctly predict the minority class.

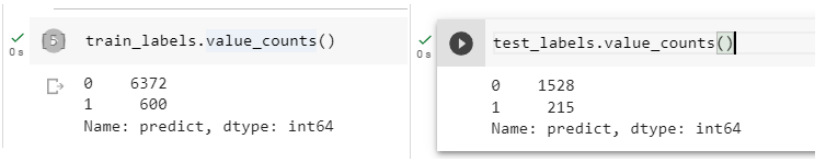


Figure 4: *Unbalance in the variable to be predicted. "0" if not viral "1" if viral*

so even if we specify in our database partitioning that we want it to be "stratify", the unbalance still exists.

One way to correct this problem is to use oversampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique).

SMOTE is an oversampling technique that creates new synthetic examples for the minority class by interpolating existing examples. Instead of simply duplicating existing examples, SMOTE generates synthetic examples that are linear combinations of adjacent examples. This allows for the creation of examples that are more representative of the minority class and reduces the likelihood of overfitting.

In the context of our machine learning project, SMOTE has been used to address the problem of class imbalance in the predicted variable of virality. To do this, SMOTE has been applied only to the training set and a number of synthetic examples have been generated for the minority class, with the aim of equalising the number of examples in both classes. This has allowed us to train a more accurate and balanced model that can more accurately predict the minority class.

2.4 Creation of the model

Once we have collected and analysed our data, we can proceed to the creation of the YouTube viral comment ranking model. First, a split of our data into two sets will be carried out: training and test.

The training set will be used to fit the model, while the test set will be used to evaluate the model's performance on previously unseen data. First, algorithms such as logistic regression, AdaBoostClassifier, MultinomialNB, ExtraTreesClassifier, RandomForestClassifier, XGBClassifier have been tested and all of them achieved an overfitting result. Second, I decided to use a classifier using BERT, however, I found that it exceeds the computational capacity offered by both my local graph and google colab, encountering the following error when doing a fine-tuning.

> "OutOfMemoryError: CUDA out of memory. Tried to allocate 300.00 MiB (GPU 0; 14.75 GiB total capacity; 13.68 GiB already allocated; 104.81 MiB free; 13.87 GiB reserved in total by PyTorch) If reserved memory is >> allocated memory try setting max split size mb to avoid fragmentation. See documentation for Memory Management and PYTORCH CUDA ALLOC CONF"

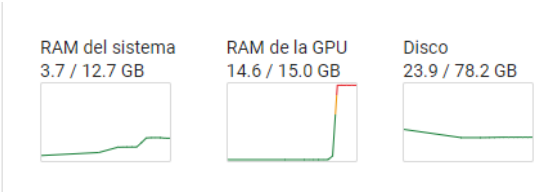


Figure 5: *You can see how the GPU memory is saturated.*

Therefore, instead of paying to use an AWS service to get resources to train my model, I decided to create a neural network of the RNN type.

RNNs (Recurrent Neural Networks) are a type of neural network specifically designed to work with sequential data, such as text, audio or time series. One of the main advantages of RNNs is their ability to capture contextual information and the temporal relationship between data.

There are several types of RNNs, including SimpleRNN, GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory). Each has its own characteristics and is used for different types of problems.

In this case, not having enough resources to pay for a cloud service such as AWS, we opted to use the RNNs available in Python libraries, such as Keras and TensorFlow. This allowed the model to be trained using the resources available on the local computer, avoiding the additional costs of using a cloud service.

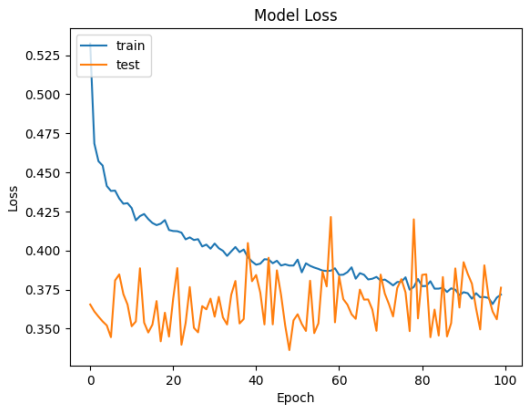
It is important to note that while using RNN may be a cheaper alternative, it may not achieve the same efficiency and scalability as using a cloud service. Therefore, it should be carefully assessed whether this option is suitable for the project in question.

Finally, the model that generalises for my training data and behaves under conditions for my test data is a neural network based on SimpleRNN. One might expect it to be LSTM as it is a more powerful recurrent neural network structure, however for this particular problem, we have comments that have few words and LSTMs are designed to handle long and complex data sequences, and if the data is too short, the additional layers of an LSTM, based on input gates, forgetting gates and output gates, may not be necessary. If the task at hand does not require long-term memory, a SimpleRNN may be more effective than an LSTM.

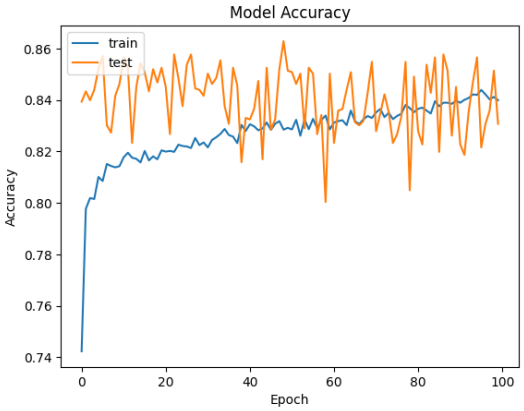
2.5 Model validation

The plots resulting from the SimpleRNN-based model are shown. The rest of the learning curves, including the LSTMs, showed cases of overfitting, so they are omitted.

In machine learning and artificial intelligence, accuracy and loss are two important metrics for evaluating the performance of a model. Accuracy measures the model's ability to correctly predict the labels in the test data, while loss measures the difference between the labels predicted by the model and the actual labels in the test data.



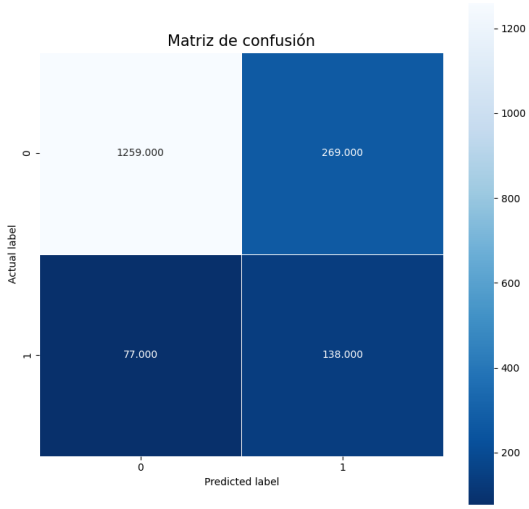
(a) Learning curve for loss.



(b) Learning curve for precision.

In particular, it is important to note that the accuracy plot shows a steady increase in accuracy as the model is trained, which is a promising sign that the model is improving over time. In addition, the loss plot shows a steady decrease in loss, indicating that the model is minimising the difference between the predicted labels and the actual labels in the test data.

We can also see other relevant estimators when doing classification, such as the confusion matrix.



(a) Confusion matrix

```
from sklearn.metrics import precision_score, recall_score,f1_score

precision = precision_score(test_labels, predictions)
recall = recall_score(test_labels, predictions)
f1 = f1_score(test_labels, predictions)


print(f'Accuracy: {round(precision*100,2)}%')
print(f'recall: {round(recall*100,2)}%')
print(f'f1_score: {round(f1*100,2)}%')
```

```
Accuracy: 36.81%
recall: 55.81%
f1_score: 44.36%
```

(b) Precision, recall y f1-score

we can see that your model correctly predicted 1259 cases where the actual label was 0 and the model also predicted 0. This is in the top left corner of the matrix. Also, the model incorrectly predicted 269 cases where the actual label was 0, but the model predicted 1. This is in the upper right corner of the matrix. On the other hand, the model incorrectly predicted 77 cases where the actual label was 1, but the model predicted 0. This is in the bottom left corner of the matrix. Finally, the model correctly predicted 138 cases where the actual label was 1, and the model also predicted 1. The confusion matrix indicates that the model has a high performance in predicting cases where the actual label is 0, as it correctly predicts most of these cases (1259) and has only a small number of false positives (269). However, it seems to have problems in correctly predicting cases where the true label is 1, as it has a significant number of false negatives (77) and a smaller number of true positives (138). It is important to note that this is only a preliminary assessment of the model's performance and that more information would be needed to make a more complete evaluation as in the test data we have unbalanced classes.

An important fact is when the classes are unbalanced, there are estimators that are more significant than the accuracy, one of them is recall where we observe a 55.81%. Recall is a metric where we measure the sensitivity of our model to predict the proportion of true positives that have been correctly identified. It is a useful metric when we are more interested in avoiding false negatives. In this case, false negatives would be when a comment that is viral is detected as non-viral. While this number may seem low, it does open the door to creating a better model that more accurately predicts when a comment will go viral or not.

2.6 Model implementation

Keep in mind that you are entering the tokenised comments, so if you try to enter a text string with your comment, it will result in an error. In addition, the sentiment is also entered for both BERT and nltk, so you would also have to get this information from the text string. For this we have made the code that can be found in my GitHub.

The function "frase a predecir" has been created such that, the final result will return the probability that the comment will go viral.

```
texto = 'dale like para que ibai participe en la siguiente velada'

frase = frase_a_predecir(texto)

model.predict(frase)
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
1/1 [==============================] - 0s 51ms/step
array([[0.31621855]], dtype=float32)
```

Figure 8: *Predicted virality of the comment to be liked*

Once you have trained the model, you can download this model with libraries such as Joblib or pickle and insert it into a simple Streamlit page, as using this function, it should be simple to do.

3 Conclusion and future steps

Possible improvements to this model would be to perform text classification based on BERT. [2]. However, I was limited using BERT by GPU. BERT is a large model with millions of parameters, and its re-training requires a large amount of processing power and memory. Therefore, it is recommended to use at least one GPU with at least 12 GB of memory to train medium-sized BERT models on moderate-sized datasets. For larger datasets or larger models, a GPU with 16 GB or more of memory may be required, which even using Google Colab may not be available. Also the use of AutoGPT

[5] [4], to do sentiment analysis using their "text-davinci-002" API in the case of sentiment analysis. However, I am also limited by the number of API calls I can make.

Further improvements to the model would be to get more viral comment data by doing some history. For example, not only based on the comments of "La Velada del Año", but to have all the history of its YouTube channel, since the comments would give us information on all the traffic and audience it has. Thus, it would have more viral comments and would give us information on what type of comment is the most searched for.

Another possible improvement would be to perform more training with different sets of hyperparameters, with the aim of finding the optimal combination of parameters to improve the performance of the model.

In summary, this comment sentiment analysis methodology could be useful for any advertising or marketing strategy involving the presentation of product or service reviews. Rather than simply displaying the most popular reviews, this methodology allows you to identify the reviews that are most relevant and valuable to your audience. Moreover, it can be especially useful in the first hours after the publication of a review, when there are not yet many votes, to sort and optimise the comments that are displayed. Once the time has passed, it can also show those comments with a high tendency to go viral and the new comments not be forgotten in order to provide a better user experience. Finally, the application of this methodology can significantly improve the effectiveness and efficiency of the presentation of reviews and comments in an advertising strategy.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[2] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Brew, J. (2019). Hugging Face's Transformers: State-of-the-art Natural Language Processing. *ArXiv preprint arXiv:1910.03771*.

[3] Bird, S., Klein, E., Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

[4] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33.

[5] OpenAI. (2021). *Introducing AutoGPT: A Framework for Automatic Architecture and Hyperparameter Search for Transformers*. Recuperado el 11 de mayo de 2023, de https://openai.com/blog/introducing-autogpt/.

[6] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.