

Sentiment analysis for football tweets in Spanish

Diwa Vilà
diwvi456@student.liu.se
TDDE16 - Text Mining

Abstract

This study aims to classify the correct sentiment given a tweet about football in the Spanish language. It compares two different methods: a Naive Bayes classifier and a BERT transformer trained on a Spanish corpus. Initial research showed limitations when performing NLP tasks in Spanish and the studies made with the same type of data (tweets) did not achieve the best results. The data consisted in 20k labeled (neutral, positive, negative) tweets from the football match between Spain and Germany during the world cup. After comparing the models, BERT performed better than the Naive Bayes approach although not by a very significant difference and some limitations were found when doing NLP in Spanish instead than English.

Introduction

Nowadays, social networks are plenty of opinions, reviews or rating generated by users. These sentiment content can be about anything and are very interesting to know the population better, the general opinion of the people attending a play, watching a movie, what they liked the most, what they disliked the most and, thus, it becomes very beneficial not only for business but also to understand the human being.

This project tries to match football with the general opinion of the viewers. During a football game, a lot of users express their opinions on Twitter about how their favourite players are performing. Thus, the aim of the project is to be able to analyse the sentiments of those fans during a football match. So, we will build a model that is able to predict whether a tweet is positive, negative, or neutral. This way, the project could be continued with the analysis of those predictions.

Emotion analysis of messages using NLP is a difficult task, and the challenge grows when the messages are not written grammatically correct or use words that do not appear in the dictionary, which is the case of tweets. This already pronounced challenge grows even more when the language used is not English, but Spanish as although being one of the most spoken languages, it has not been as addressed as in the English context.

Theory

Natural Language Processing is a field of Artificial Intelligence focused on the written text. The goal is for computers to understand human language. We can use NLP to perform Sentiment analysis to analyse text to determine emotional tone, attitude or sentiment expressed. In this project we will use two main methods to extract relevant features with the final objective to classify text into positive, negative or neutral sentiment. In this project we will define sentiment as a personal positive, negative or neutral feeling.

Multinomial Naive Bayes

It is a probabilistic Machine Learning algorithm that applies the Bayes theorem for multinomially distributed data. The Bayes theorem is a fundamental principle in probability theory and makes a simplifying assumption of feature independence given the class label. It represents a way to compute the probability of event A happening given that event B has occurred:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where A and B are events and $P(B) \neq 0$. $P(B|A)$ and $P(A|B)$ are conditional probabilities and $P(A)$, $P(B)$ are the probabilities of observing A and B without any given conditions (marginal probability).

The Naive Bayes algorithm makes a big assumption, the naive assumption. It comes from assuming that the features are conditionally independent given the class label. This means that the presence of a features does not influence other features. However, the model is often used in cases where the “effect” variables are not actually conditionally independent given a cause variable. Even when the conditional independence is not true, the Naive Bayes model can work well, especially with a large number of features.

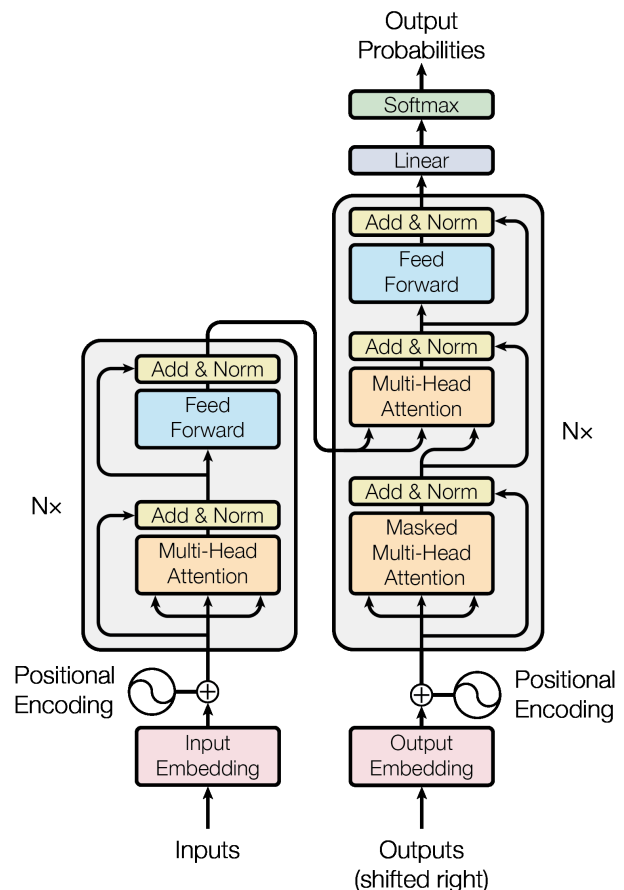
Multinomial Naive Bayes is commonly used in Natural Language Processing tasks like text categorisation, sentiment analysis or spam detections. It is a very simple and efficient model, which make it a popular choice for those classification problems.

Transformers, BERT

A Transformer is a deep learning architecture introduced in 2017. It revolutionised the field of natural language processing as it addressed the limitations of previous models and significantly improving the efficiency and effectiveness of NLP tasks.

The principal limitations of the previous models were: They did not have an inherent order of notion (1), there were no nonlinearities added for deep learning (2), they used weighted averages and they looked in the future, which we want to prevent (3). Problem 1 was solved with positional embeddings, problem 2 with a Feed Forward Network and problem 3 with masked attention. This third one is why this architecture is known. Attention allows to capture relationships between words in a sentence regardless of their distance, in contrast of earlier models that relied on recurrent or convolutional layers that struggles with long-range dependencies. A transformer architecture looks like:

The Transformer used in this project is BERT (Bidirectional Encoder Representations from Transformers) has has a profound impact on MLP tasks, including sentiment analysis.



Data

Given that the world cup was celebrated during the month of November and December, we extracted tweets with the tweeter API during a football match. The data extracted correspond to the Spain - Germany game celebrated on the 27th November which finished 1-1. From this match we extracted tweets that had the name of a player of the Spanish team, we did not extract data from just one player, we extracted from all. Another constraint we put when extracting the data was that the tweet had to be in Spanish, as the goal was to do a sentiment analysis of the Spanish fans (more likely to be spaniards and tweet in Spanish).

The data was extracted with the Twitter API for Python “tweepy”(<https://www.tweepy.org/>). To extract the data, we created a function that we executed during the match time. During the time it was executed, it extracted the id, the time it was created, the text, the reply id, and the quote id from all tweets in Spanish that contained at least a name from a Spanish player and wrote them in the output. The output consisted of a json file for each of the players, if a tweet mentioned more than one, it will

appear in more than one json file. Once this json file is created, we converted it to a .csv adding the column of the player's number. When we had the .csv created (one for each player), we tagged the data being positive, negative or neutral. The data was tagged regarding the opinion of the player referenced, not the overall tweet (as the objective is to make sentiment analysis of the players). So, if a tweet was: *“El partido no me ha gustado nada pero Pedri ha jugado muy bien”* the tag will be positive as although the user did not like the match (negative feeling), it has a positive opinion about the player referenced.

Then, we got all the tagged data and merge it together into one .csv, which is the final corpus we will work with. **This corpus has 21.109 tweets and 7 characteristics:** id, created_at, text, tag, reply_id, quote_id, player.

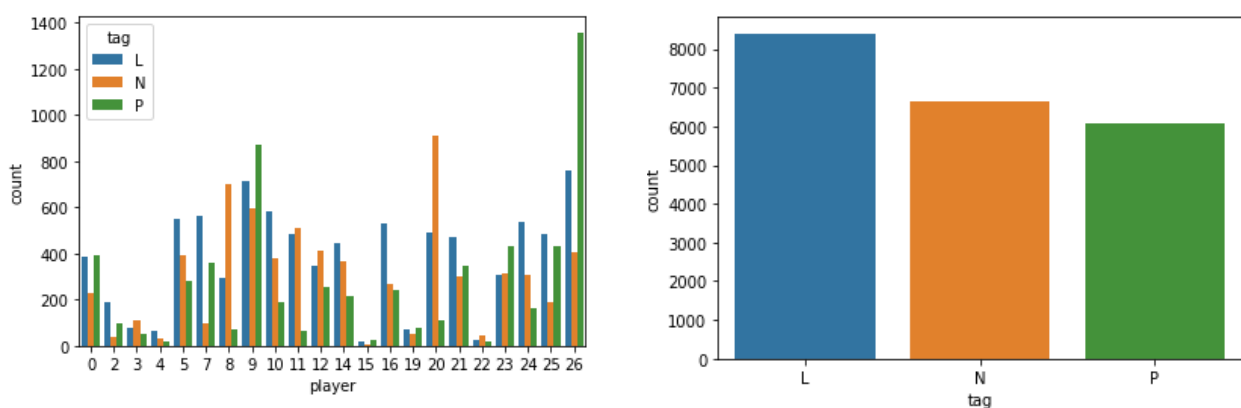
- **id:** the primary key of the data, unique for each tweet
- **created_at:** the time which the user tweeted
- **text:** the raw tweet
- **tag:** [P, N, L] positive, negative neutral. the sentiment of the tweet
- **reply_id:** if the tweet is a reply from another tweet it will have the id of that tweet, if not, it is null
- **quote_id:** if the tweet is quoted by another tweet, its value will be the id of that tweet, else it is null
- **player:** the (dorsal) the player the tweet makes reference to. For instance, Gavi has number 9 and the tweet is about Gavi, it will have the value 9. If the tweet makes reference to more than one player, this tweet will appear more than once, one for each player it makes reference. This way, if a tweet is positive for one player but negative for another it will be reflected. See the following example:

“Por favor, que mal Carvajal, Azpilicueta tiene que jugar siempre”.

The tweet is negative for Carvajal (20) and positive for Azpilicueta (2). In the corpus this will be written the following way:

id	text	tag	player
1	Por favor, que mal Carvajal, Azpilicueta tiene que jugar siempre	P	2
1	Por favor, que mal Carvajal, Azpilicueta tiene que jugar siempre	N	20

After all, we made some visualisations of the data:



In the left plot, we see all the players and the positive, negative and neutral tweets. We see that some players like Pedri (26) have a larger amount of positive tweets and others, like Carvajal (20) have a large number of negative opinions. In the middle we have the total of the opinions of the users: most of the tweets are neutral and the positive and negative tweets are close in volume. However there are more negative than positive opinions.


Method

To carry out this study, we used python and the method was the following: Pre-processing the data, training model + fine tuning model and final model

Pre-processing the data:

We converted out raw data into “cleaner” text with the following methods and techniques:

- **Remove URLs.** Many tweets contain URLs to websites or other tweets but they are not relevant for our task. We removed them as they only make the text noisier.
- **Convert numbers (1,2) into text (one, two).** Numbers were both written in numbers or text, we decided to unify them all into text. Spacy, one of the main Python libraries to do preprocessing does not work in Spanish, so we used the library num2words.[NUM]
- **Convert all words to lowercase.** In twitter, some users write following grammar rules but generally social media users write in a colloquial way. One thing they do sometimes is writing without capital letters, so to unify all text, we converted all words to lowercase.
- **Remove stopwords.** Stopwords do not provide any useful information. In this case, we did not delete the words from a list of stopwords. Instead we used the Spacy library to get the type of word and then deleted :ADP, AUX, DET, INTJ, PART, PRON, PROPN, SYM. Those are the main types of word that do not provide key information.
- **Remove punctuation marks:** We removed punctuation marks as they do not provide any emotion by themselves, they emphasize an emotion already expressed.
- **Return list tokens**

Tweet	Preprocessed text
Que buena definición de Morata y el pase de lujo de Jordi. Como no disfrutar del buen juego.	['que', 'buen', 'definición', 'morata', 'y', 'pase', 'lujo', 'jordi', 'como', 'no', 'disfrutar', 'buen', 'juego']
HORA DE ALEJANDRO BALDE	['hora', 'alejandro', 'balde']
@Nererodrig Lamentable koke, que nivelito 	['@nererodrig', 'lamentable', 'koke', 'que', 'nivel']

Baselines and evaluation methods.

- **Accuracy:** How often a classification model is correct overall. It shows how often a model is correct when predicting the target class. **$Precision = TP / (TP + FP)$**
- **Recall:** It represents how many relevant items are retrieved, so it is the fraction of the true positive predictions and the relevant elements (true positive + false negatives) **$Recall = TP / (TP + FN)$**
- **F1 score:** balances the precision and recall, as if we have non-balanced classes (for instance we have two classes and 99 observations are from class A and 1 observation from class B) the precision and recall may not reflect correctly the performance of the model (if we have a model that just predicts class A, it will have a precision of a 99%). **$F1 = 2 * Precision * Recall / (Precision + Recall)$**

After preprocessing the data, we executed a couple of **random baselines**: Stratified and Most frequent.

Both baselines are from the *sklearn* dummy classifier class library. The Stratified baseline randomly samples one-hot vectors from a multinomial distribution. The most frequent always returns the most frequent class label, in this case ‘neutral’.

The next step is training the models and fine tuning its parameters:

Multinomial Naive Bayes. To do this model, we have to vectorize the text. Here, we will use the count vectorizer as we have specific data and user profile. After vectorizing the tweets in the training data, we instantiate the Naive Bayes model with the default parameters and evaluate it on the test data. *The results are shown in the result section.*

After computing the model with the default hyperparameters, we performed a grid search to find the best settings for our model. We did a grid search and not a manual change of the components as it is easier and less exhaustive. The grid search allows to run an exhaustive search for the best hyperparameters over a grid of possible values. As the hyperparameters can not be tuned on the test set, we will use a 5-fold-cross-validation with the following hyperparameters:

- in the vectorizer, we tried a binary model in addition to the default bag-of-words
- In the vectorizer we also tried extracting bigrams in addition to unigrams (two possible parameter values)
- In the Classifier, we used additive smoothing with values 1, 0.1 and 0.5.

The best set of parameters was to use additive smoothing with value 0.5, use bigrams and the default bag of words. We also run this model.

Then, as our data was not balanced, we had more neutral observations, we balanced out data using an under sampler. We choose to remove data from the classes with more observations as the class with less observations had enough tweets for the model to perform correctly, we were not losing substantial information. If it were not the case and reducing our data would lead to a huge loss of information, we would have to choose another method. After balancing the data, we ran again the default Multinomial Naive Bayes model with the new dataset.

BERT Transformer. To do this first we had to create our corpus and tokenize our sentences, which returned a dictionary of input ids and attention masks. Then we created attention masks to our tokens and converted to tensors and divided the data into train and test. We used a pre-trained BERT, in this case *dcycgile/bert-base-spanish-wwm-uncased* which is a package from Hugging Face transformers library compatible with pytorch. It is called BETO, a BERT model trained on a big Spanish Corpus. Then we just finetuned the optimizer (Adam) weight decay parameters: we used bias, gamma and beta and trained with the three different and got the best result. We did the same two times, with 0 and 0.5 decay. Finally we trained the model and computed the accuracy, recall and f1-score.

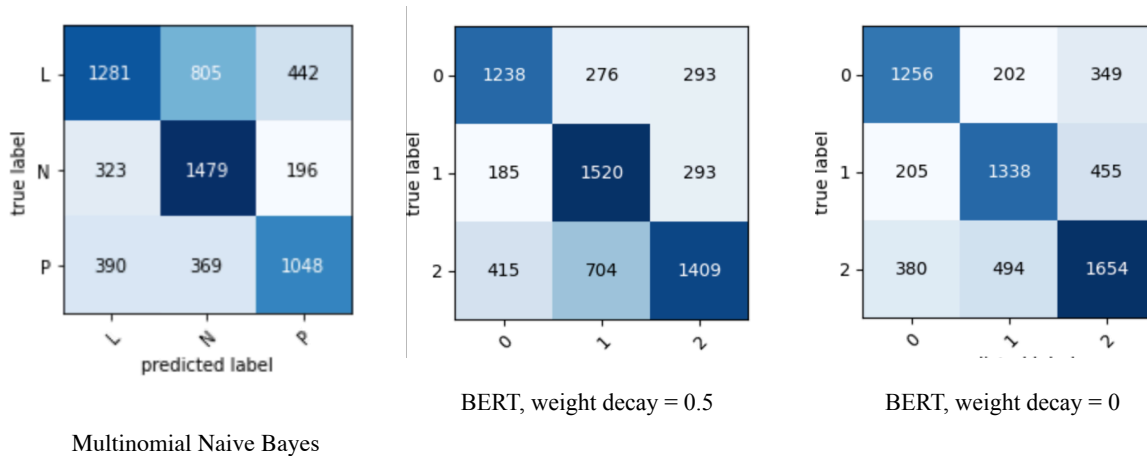
Note that the Transformer is much efficient when ran with a GPU rather than a CPU. In this case, the code is ran with NVIDIA GeForce RTX 3060 Laptop GPU and it may not be able to run in CPU (the corpus size should be reduced). For it to run in GPU, CUDA has to be installed.

Results

	Accuracy	Recall	F1-score
M Naive Bayes (default)	0.61	0.60	0.60
M Naive Bayes (balanced)	0.62	0.60	0.59
M Naive Bayes (best hyperparameters)	0.62	0.61	0.61
BERT (0 weight decay)	0.66	0.67	0.67
BERT (0.5 weight decay)	0.66	0.65	0.65
Random Baseline			0.34
Most frequent Baseline			0.39

As we see, the models perform better than the baselines. There is little difference between the Naive Bayes classifier and its variants and the BERT and the different weight decay. We can also say that although the difference is not sustainable, the BERT model performs slightly better than the Naive Bayes approach.

We also computed the confusion matrix for the default Naive Bayes, and the two BERT models



*** In the BERT models, 0 = P, 1 = N and L = 2 (L stands for neutral)*

We see that in two cases, the Naive Bayes and the BERT with weight decay the best classified class is the negative one, whereas in the BERT without weight decay (weight decay 0) classifies best the neutral tweets. We also see that in Naive Bayes, a significant number of cases from class neutral, are predicted as negative, which is not a good sign. In BERT with weight decay, something similar happens, in this case a significant number of negative observations are classified as neutral. Overall, the BERT without weight decay seems to be the best model regarding the confusion matrix and accuracy, recall and f1-score.

Discussion

The three models performed significantly better than the baselines, which had an f1-score that did not even reach 0.5. However the best model did not even reach an f1-score of 0.7, which also not a good outcome.

In addition, the best model was a BERT model, which takes much more resources of GPU and time than the Naive Bayes model but the increment in accuracy was not that much, not even a 0.1. This makes us think the following problem: is the amount of extra resources we use for the small increase in accuracy worth it? In this case, I would say yes, as the Naive Bayes model is closer to a 0.6 f1-score whereas the BERT model is closer to 0.7.

Although the results may not seem the best, when we compare to other work done on classification / sentiment analysis of Spanish tweets, we see that various studies reach an accuracy of 0.6, 0.65. This means that our study is not that far away from the work done in the field. However, when we compare the results to related work of sentiment analysis done with English tweets, the accuracy is much higher, reaching a 0.8-0.9.

Bearing all of this in mind, we reach the conclusion that language is a significant barrier when performing NLP, as some languages grow faster and have much more resources than others, in this case Spanish. The resources drop when doing NLP in another language and the ones available are more rudimentary and not get the best accuracy when preprocessing data or creating a Transformer. This highlights the importance of a good knowledge of the language and the basics before performing

more high-end tasks like sentiment analysis. For example, when preprocessing the text there were little libraries available.

Regarding this project maybe if the preprocessing of the text were to be more exhaustive, not only using libraries but also doing some manual preprocessing, the remaining text would be more valuable. For instance, knowing that the language in twitter is colloquial and users tend to use abbreviations of words, it could have been a good approach to replace emojis for the emotion they evoke as they are remarkably important but they do not facilitate the process. Another thing that could be done are to replace common abbreviations into their correct form like *q* (*que*).

Conclusion

NLP models for sentiment analysis is a challenging task as humans have very different and ambiguous ways to express their emotions and when expressed in text, social context is also important. In consequence, it is not only important to have a large amount of data but also having high-quality data and a good understanding of the language.

In the project, we compared different approaches to predict the sentiment of tweets about football in Spanish and got a result of a 0.67 accuracy, which means that it classifies emotions correctly more than half the times.

As future work, the model could be improved by enhancing the data, in the preprocessing phase as discussed in the discussion section and in the Transformer model, where we used a pre-trained model. It could be a good experiment to try and Train a Transformer from scratch as the language is very task specific (colloquial, about football and a limited number of players).

References

1. de Arriba, A., Oriol, M. Franch, X. Applying Sentiment Analysis on Spanish Tweets Using BETO
2. Zhag, L., Wang, S. Liu, B. (2018). Deep Learning for sentiment analysis. A survey.
3. Vaswani et al (2017). Attention is All You Need
4. <https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased>
5. Class Notes
6. Code in GitHub repository: https://github.com/diwavila/Text_mining