# Text Analytics project – Sarcasm detection

Giuseppe Mirko Milazzo

October 18, 2022

## 1 Introduction

The definition of "sarcasm" is :"the use of irony to mock or convey contempt".

Sarcasm detection is a very restricted research field in NLP, it is a specific case of sentiment analysis where instead of identifying the sentiment in the whole spectrum, the focus is on sarcasm.

The file `Sarcasm_Headlines_Dataset.json` contains a total of 26709 records of different journal's headlines, each observation is composed by the *article_link*, the *headline* of the news and the label *is_sarcastic*: I decided to drop *article_link* and to focus exclusively on *headline*.

The goal of the project is to understand whether an headline is sarcastic or not.

In order to achieve this, I decided to extract some "basic nlp count based features", leveraging text sentiment and perform some cleaning steps to obtain a *clean_headline* on which apply a "Clustering-based Topic Modelling" for inspect and analyze main topics from the headlines. Given the different nature of the response variables I decided to proceed with the classification followed by different global explainability approaches on the models obtained to understand the logic behind the black boxes.

The report is organized in different sections. In Section 2 all the preprocessing steps for prepare the data for further analysis are explained. Then, in Section 3 through a "Clustering-based Topic Modelling" approach I tried to inspect the data based on their label (so when *is_sarcastic*=0 and when it is 1) to understand whether the feature space could be naturally separated in groups of observations, i.e. whether clusters of different nature exist and if they are somehow coherent since they belong to the same label, and eventually, perform LDA (LatentDirichletAllocation) on each cluster, in the whole dataset with parameter *n_components*=*k* (where k is the number of clusters) and in the whole dataset with *n_components*=1 to obtain a "General Topic", in the end we have the comparison of the results. At this point I proceeded with the classification task in the Section 4 and with the explanation of the results in the Section 5. Some possible future implementation are mentioned in the conclusion.

## 2 Preprocessing

The dataset contains 26709 instances and 3 features, it is almost balanced: 56% with *is_sarcastic*=0 and 44% with *is_sarcastic*=1. As already said, I decided to focus exclusively on *headline*.

First I started creating some "basic nlp count based features", that can be appreciated in the table below.

| **Feature** | **Type** | **Meaning** |
|---|---|---|
| *char_count* | integer | number of characters |
| *word_count* | integer | number of words |
| *word_density* | float | number of characters /number of words+1 |
| *punctuation_count* | integer | number of non-words |

Table 1: List of count based features extracted from *headline*

Then with VADER (Valence Aware Dictionary and sEntiment Reasoner) which is a lexicon and rule-based sentiment analysis tool it was possible to leverage sentiment analysis in order to extract 4 new features, while the *Subjectivity* of each instance was extracted with TextBlob. The features extracted and their meaning are in Table 2.

| **Feature** | **Type** | **Meaning** |
|---|---|---|
| *neg* | float | prob. of sentiment to be negative |
| *neu* | float | prob. of sentiment to be neutral |
| *pos* | float | prob. of sentiment to be positive |
| *compound* | float | normalized score between -1 and 1 |
| *Subjectivity* | integer | amount of personal opinion |

Table 2: List of sentiment based features extracted from *headline*

At this point I focused on *headline* where several steps were performed: lower the text, fix of contractions, remove of unnecessary characters and lemmatization to obtain a new feature called *clean_headline* on which run the TF-IDFVectorizer, a module that compute the word counts (tokenization), idf and tf-idf values.

## 3 Clustering-based Topic Modelling

The aim of this section of the project is trying to understand the main subjects of both sarcastic and not sarcastic headlines, to achieve this, I splitted the entire dataset with respect to the target values, computed the TF-IDFVectorizer with: *min_df*=5 (building the vocabulary it ignores terms that have a document frequency strictly lower that 5), *ngram_range*=(1,2) (considering both unigrams and bigrams) on each subset so that I obtained 14985 rows with 4256 features for target 0 and 11724 rows 3620 columns for target 1; during the process I removed a certain number of stop words in order to achieve better results (the list of stop words have been tuned after several attempts); finally I performed the clustering analysis. I implemented a K-Means algorithm maintaining a low number of iterations (10 iterations) using the Euclidean distance. In Figure 1 and Figure 2 we can observe the 2 *Elbow plots*.

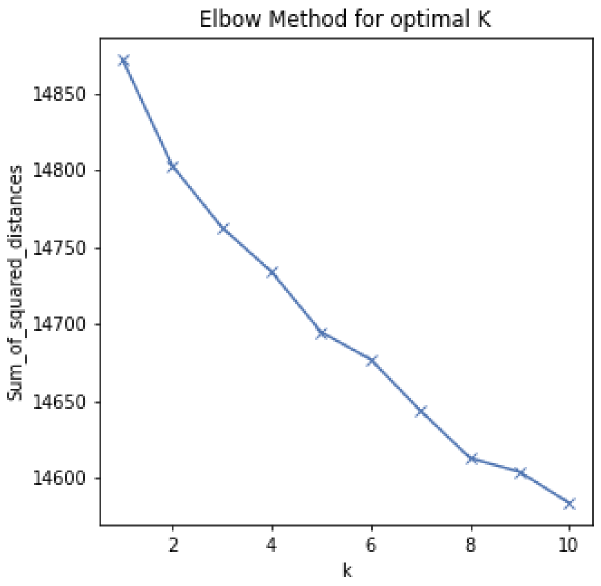

Figure 1: Elbow plot, *is_sarcastic*=0, k-Means clustering, Euclidean distance and SSE error

Seems evident that there is not a clear k to choose, nevertheless I proceeded with $k$=5 for not-sarcastic headlines and $k$=4 for sarcastic ones. We can appreciate the distribution of each cluster in both samples in Table 3.

| **Clusters_0** | **Size** | **Clusters_1** | **Size** |
|---|---|---|---|
| *0* | 11175 | *0* | 9243 |
| *1* | 155 | *1* | 396 |
| *2* | 1636 | *2* | 1138 |
| *3* | 1365 | *3* | 947 |
| *4* | 654 | | |

Table 3: List of clusters extracted from subsets of different labels and their size

From Table 4 it is interesting to notice how the results obtained running topic modeling on each

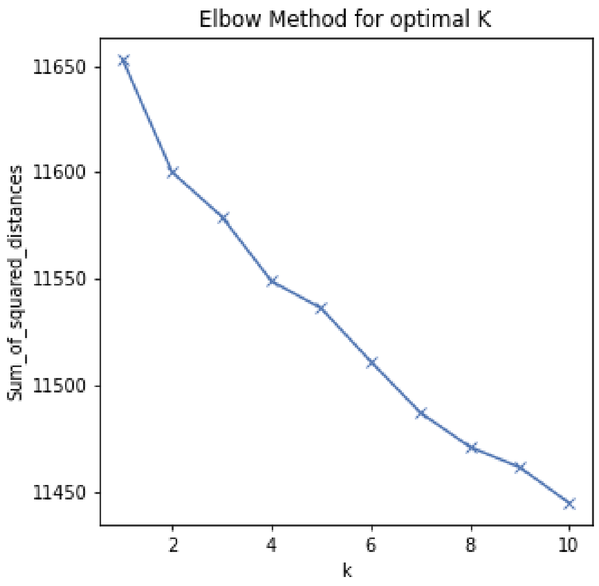

Figure 2: Elbow plot, *is_sarcastic*=1, k-Means clustering, Euclidean distance and SSE error

cluster are overall more coherent than those where *n_components=k* (see Table 5), as we can observe in the latter the word "trump" is spread in all topics, while applying LDA on each cluster it is present only in 3 out of 5 clusters, also, despite all topics are about politics, we can distinguish that *Topic Clust 3* focuses on Trump administration, while *Topic Clust 1* on different USA politicians.

About sarcastic headlines, observable in Table 6 and Table 7, the topics extracted from clusters are, again, relatively more coherent than those extracted with *n_components=k*: in the latter the word "man" is spread in all topics, while applying LDA on each cluster it is present only in *Topic Clust 1* and *Topic Clust 2*. I did not found topics related to politics (like from topics of not-sarcastic headlines), the coherent subjects are about family (*Topic Clust 0*) and women (*Topic Clust 1*). Finally, topics extracted with LDA using *n_components=k* performs poorly, not capturing the different subjects hidden in the headlines, *General Topic* instead seems to capture both family and women topics.

| **Topic** | **Words** |
|---|---|
| *Topic Clust 0* | woman, man, way, people, obama, watch, american, black, call |
| *Topic Clust 1* | hillary, clinton, hillary clinton, trump, donald trump, donald, clinton win, sander, bernie sander, bernie |
| *Topic Clust 2* | good, thing, need, love, want, life, know, like, kid, change |
| *Topic Clust 3* | trump, donald, donald trump, president, gop, call, ban, would, administration, attack |
| *Topic Clust 4* | new, york, new york, show, trump, book, film, video, new book, new album |

Table 4: List of topics extracted from each cluster, not-sarcastic *headlines*

| **Topic** | **Words** |
|---|---|
| *Topic 0* | world, trump, live, lesson, week, look, like, new, violence, people |
| *Topic 1* | trump, donald, donald trump, clinton, hillary, hillary clinton, president, obama, gop, sander, bernie |
| *Topic 2* | trump, new, woman, man, mom, help, york, new york, american, america |
| *Topic 3* | new, trump, change, need, know, way good, life, thing, white |
| *Topic 4* | trump, care, woman, police, shoot, health, talk, kid, republican, court |
| *General Topic* | trump, new, woman, donald, donald trump, man, good, way, thing, world |

Table 5: List of topics extracted with *n_components=k*, not-sarcastic *headlines*

| Topic | Words |
|---|---|
| *Topic Clust 0* | nation, american, make, study, child, trump, family, good, obama, back |
| *Topic Clust 1* | woman, man, feel, make, think, self, like, pregnant, woman know, man woman |
| *Topic Clust 2* | man, local man, think, life, local, make, like, no, good, look |
| *Topic Clust 3* | like, guy, clinton, introduce, give, leave, release, put, supreme court, supreme |

Table 6: List of topics extracted from each cluster, sarcastic *headlines*

| Topic | Words |
|---|---|
| *Topic 0* | man, white, study, woman, house, dad, call, white house, make, nation |
| *Topic 1* | man, school, die, life, high, no, every, paul, discover, nation |
| *Topic 2* | man, make, woman, show, announce, think, use, kill, nation, american |
| *Topic 3* | like, woman, look, local, dead, american, even, fuck, couple |
| *General Topic* | man, woman, nation, make, american, study, like, good, child, family |

Table 7: List of topics extracted with *n_components*=*k*, sarcastic *headlines*

## 4 Classification

For the supervised task I first performed a stratified split with *test_size*=0.2, doing so I obtained a training set of size (21367, 11) and a test set of size (5342, 11), then I implemented a TF-IDFVectorizer with *min_df*=3 and *ngram_range*=(1,2): this task was applied to *clean_headline* for both train and test set. Given that the considerable amount of features (around 10600) could lead to curse of dimensionality, while also making training time unfeasible, I applied an intermediate step of feature engineering using a SelectKBest with *score_func*=chi2 (chi-square seemed the most suitable score function given the nature of data under analysis).

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Figure 3: Formula of chi-square test to obtain best features: c is the degree of freedom, O is the observed value(s), E is the expected value(s)

A chi-square test is used in statistics to test the independence of two events. Given the data of two variables, we can get observed count O and expected count E. Chi-Square measures how expected count E and observed count O deviates each other. The only drawback of SelectKBest is that *k* (number of features) needs to be set as input, this led to several attempts for obtaining the best possible results in terms of accuracy and the lowest possible level of overfitting: I decided to set *k*=4000.

The final step was to join these 4000 features from both training and test with the respective count-based features and sentiment-based features, the columns *headline* and *clean_headlines* were discarded. After the join the training and test size are respectively (21367, 4009) and (5342, 4009).

Various machine learning models were implemented, in Table 8 we can appreciate the results.

| Model | F1 | Accuracy | AUC |
|---|---|---|---|
| *Logistic Regression* | 0.83 | 0.81 | 0.89 |
| *Decision Tree* | 0.74 | 0.69 | 0.73 |
| *Random Forest* | 0.74 | 0.62 | 0.74 |
| *XGBoost* | 0.81 | 0.78 | 0.87 |

Table 8: Models recap: F1 score, Accuracy and AUC metrics

On all models I implemented a k-fold stratified cross validation with *n_splits*=4 and *n_repeats*=2 and a RandomizedSearchCV with *n_iter*=4, inserting as input different parameters, the goal was improving the results while reducing possible overfitting.

The best model is the Logistic Regression with an *Accuracy* of 0.81: this is expected since Logistic Regression scales well with the amount of data, unfortunately the high dimensionality can lead to overfitting (which is present for a slight amount), the default *penalty*=l2 applies a "ridge" regularization that helped in reducing it.

Since tree-based models generally performs poorly with high dimensional datasets, we have a low level of *Accuracy* on both Decision Tree and Random Forest, while XGBoost (also a tree-based model that uses an ensemble method as Random Forest) performed quite well with an *Accuracy* of 0.78: the implementation of the Gradient Boosting method uses more accurate approximations to find the best tree model.

## 5 Explainability

Several explainability approaches were implemented: I exploited the built-in function *feature_importances_* offered by Random Forest to obtain the features that contributed the most (see Table 9).

| Feature | Importance |
|---|---|
| *nation* | 0.125 |
| *trump* | 0.089 |
| *area man* | 0.063 |
| *punctuation_count* | 0.061 |
| *word_count* | 0.050 |
| *man* | 0.042 |
| *word_density* | 0.039 |
| *introduce* | 0.030 |
| *donald trump* | 0.024 |
| *Subjectivity* | 0.024 |

Table 9: 10 most important features in Random Forest classification

It is interesting how some of the words with the most contributions in the classification are those identified by LDA as topic.

A TREPAN was implemented on Logistic Regression to express the choice made by the algorithm as a Decision tree: TREPAN learns to predict the label returned by the Logistic Regression, not the original one in the dataset, also it returns an accuracy score which in this case is called "fidelity" and indicates how much the Decision Tree is adherent (able to mimic) to Logistic Regression. The fidelity of TREPAN with *max_depth*=20 is 0.72. The results are in the notebook: within the most important feature we find *man* and *trump* that were also the most frequent words detected during the clustering-based Topic Modeling analysis respectively on sarcastic and not-sarcastic dataset.

SHAP (SHapley Additive exPlanations) was lastly applied on XGBoost (see Figure 4), thanks to the violin plot we can observe how features are ordered by their effect on prediction and how higher and lower values of the feature will affect the result. It is clear that count-based features were crucial, also it is interesting how the contribution of "man" and "trump" to the prediction is almost symmetric, we also see that *Subjectivity* fairly contributed as well.

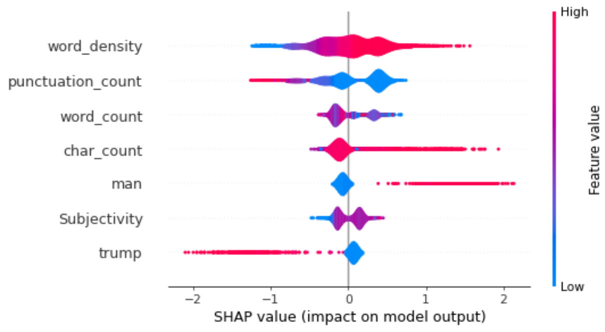

Figure 4: Violin plot on SHAP values extracted from XGBoost

## 6 Conclusion

While sarcasm detection remains one of the most challenging sentiment analysis task of NLP, a lot of improvements have been done until now. Surely data did not offer a wide variety of topics and arguments: most headlines were about politics , women situation and family, so could be interesting to see the same approach of clustering-based topic modeling per label with a way more diverse data as input to see if clearer results could be reached.

Possible future implementation could be the use of NER (Named Entity Recognition) during preprocessing steps to help in recognizing proper names. During the process I discarded the *article_link*, but this feature could be used for add important information to the supervised task: certain journals/sites usually tends to be more sarcastic than others. Moreover for the classification task, could be interesting the implementation of frameworks like BERT (Bidirectional Encoder Representations from Transformers), state of art model for text data, that should help in gaining higher levels of accuracy i.e. less errors and a more appropriate detection of sarcasm for journal's headlines.