***Abstract***

Recently, many phenomena appeared and spread in the Internet, especially with the huge propagation of information and the growth of social networks. Some of these phenomena are fake news, rumors and misinformation. In general, the detection of these phenomena is crucial since in many situations they expose the people to danger.

Journalism made several efforts in addressing these problems by presenting a validity proof to the audience. Unfortunately, these manual attempts take much time and effort from the journalists and, at the same time, they cannot cover enormous volume of fake news. Hence, there is the need for addressing the problem from an automatic perspective.

Different approaches have been proposed to address the problem. We present an approach that combines lexical, word embedding’s and n-gram, with Word2vec, feature extraction/selection and several analytical techniques such as:

* Linear Regression with weight class to balance data (Baseline Model)
* Decision-tree
* Random-Forest
* Random Forest + Bagging
* Support Vector Machine Classifier (SVM)
* XGBoost
* Naive Bayes

Our proposed approach has achieved an accurate result of (xx% ).

***The objective***

The general objective of the project is to design and implement an AI application capable of identify if an article is fake or not.

***The approach***

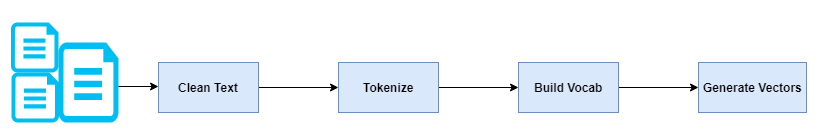
Although the general objective seems clear enough, we have to define our approach regards the problem. Here we have our first Decision. Many combinations of approaches can be used:

* Initially we deployed AI techniques to compare the title of the article with the body text identifying if they correlated to each other.
* *Secondly we identify the subject of the text and try to guess the chance that it be a fake news (For instance news about Hollywood have a much better chance to be fake than science news –Sentiment analysis)*
* *Thirdly we identify the tone of the article – Angry, scary, condescending – Fake news usually try to induce strong feeling in people.*

We are going to use a combination of these factors.

***Data preparation***

The general sequence of the data preparation will follow the sequence described in the subsequent diagram:



:

**First step – Getting the fake news database**

We crossed the two files Train\_stances and train\_bodies.csv using the Body\_ID column as the key index creating a new table called train\_cross.csv. We did the same with the two test datasets.

We are going to use two sets of data:

1. Train-cross.csv -> 49.972 rows
2. Test-cross.csv -> 25.413 rows

Fields/columns in both files:

* articleBody
* Headline
* Stance

Stance contains the information about the correlation between the articleBody and the Headline.

This is the information the AI/ML model supposed to infer. The general idea is that if the headline disagree or is unrelated with the text it probably will be fake news.

The Stance indicates the correlation between the article body and the article headline:

* unrelated 36.545
* discuss 8.909
* agree 3.678
* disagree 840

**Second step cleaning the data**

1. Remove all irrelevant characters such as any non-alphanumeric characters
2. Remove words that are not relevant, such as “@” twitter mentions or urls
3. Convert all characters to lowercase, in order to treat words such as “hello”, “Hello”, and “HELLO” the same
4. Consider combining misspelled or alternately spelled words to a single representation (e.g. “cool”/”kewl”/”cooool”)
5. Consider lemmatization (reduce words such as “am”, “are”, and “is” to a common form such as “be”)
6. Remove additional words which we consider not relevant

**Third step –Tokenized**

This phase consists in breaking the text into individual words and counting the words

Once we did that we may separate verbs, from adjectives and nouns.

**Fourth step – Create a vocabulary**

Here we separate verbs, nouns and adjectives and may use techniques like word2vec, Glove or to guess around which words the words in our text live.

**Finally vectorization**

Now we have to transform our words into numbers to allow them to be treated by the pre-existing algorithms. We can do that using some of the following techniques:

* Bag of Words
* N-gram
* TF-IDF
* Word2Vec (google)
* Glove (Stanford)
* Fastest (Facebook)

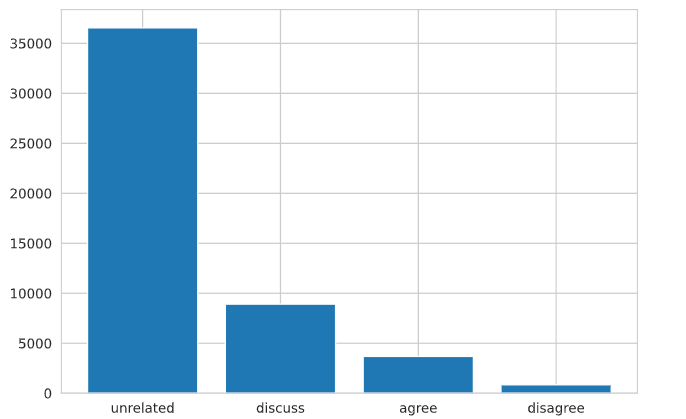
We are going to use Bag-of-words, N-gram,TF-IDF and Wrod2vec. After vectorizing using several techniques we decided to do the scenario analysis using basically two techniques (BOW and TF-IDF).

**Analytical model data preparation**



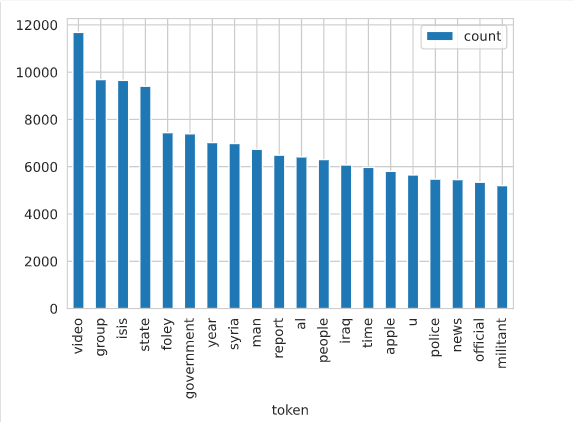
***Data exploration/Evaluating the data***

**Quantity per type of stance**

******

As the most real-life problems, our dataset has many imbalanced classes. Imbalanced data pose classification problem for predictive modelling as most of the machine learning algorithms are used for classification were designed around the assumption of an equal number of examples for each class. As a result, models train on imbalanced data have poor predictive performance specifically on minority class. We are going to explore different data balancing strategies for our dataset.

**Most frequent words**

****

**Word counting**

****

**Stance- Unrelated**

******

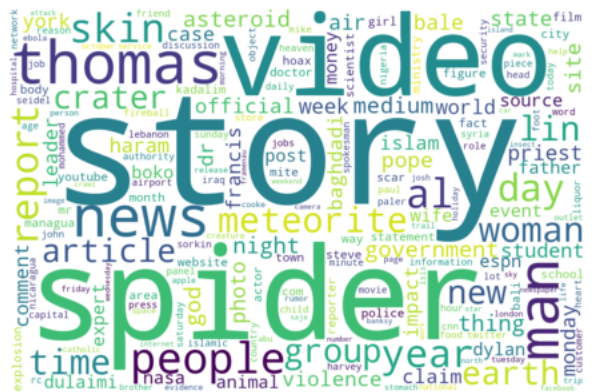
**Stance- discuss**

******

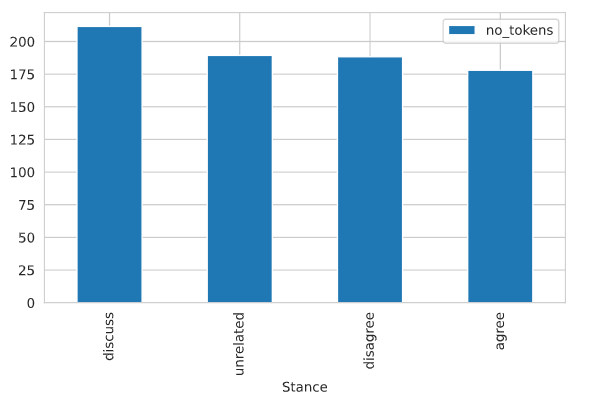
**Stance- agree**

******

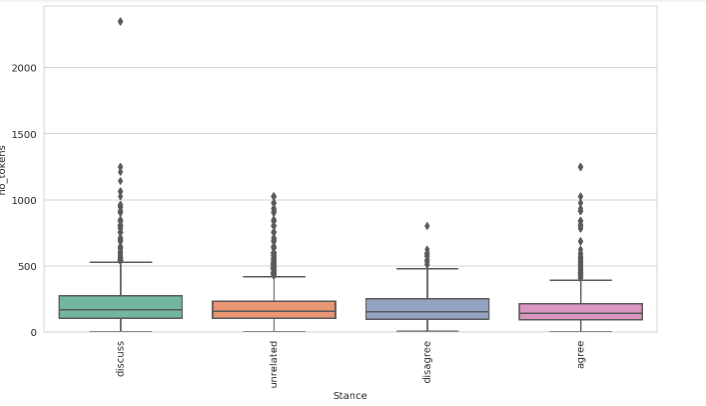
**Stance- disagree**

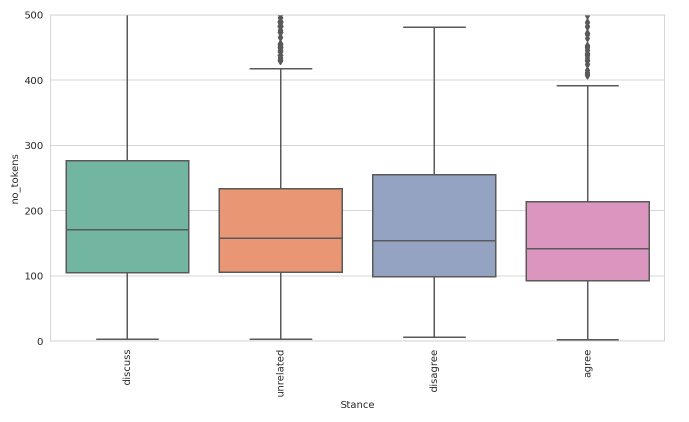
******

**Number of words per category (Cleaned)**

******

**Concentration of number of words per stance**

******

******

***Vectorization***

We decided to deploy basically two strategies to have our texts vectorized:

* Bag of Words
* TF-IDF



Due the fact that the data is unbalanced we adopted four strategies for the sample treatment. Note that the two vectorization approaches were combined with the four different sample treatments:

* No treatment
* SMOTE
* Under sampled
* Over sampled

We also combined the results with the hyper-parameters which gave us two more possible combinations:

* Without hyper-parameters
* With hyper-parameters

Therefore, each analytical technique had 16 sources possibilities 2 x 4 x 2

***Feature extraction (creating the hyper-parameters )***

We created seven synthetic variables to help us to be more assertive regards matching the headers with the article bodies:

* Headline\_sentiment
* articleBody\_sentiment
* Question\_mark
* Word\_overlap
* Cosine-similarity
* Cross
* Match
* Sent

The first two were created based on the NLP sentiment analysis, they indicate if the header and the body transmit positive or negative sentiments, we converted them into one called sent where if they match we have “1” and if they don’t we have “0”. It is our understanding that this information can be very useful given the fact that the correlation between the header and the body is key to identify if the article is fake or not, and it is much more likely that they are not correlated if they transmit different sentiments. We may compose a synthetic variable (0-1) “1” if they are the same sentiment or “0” if they are not.

The third one was created based on the existence or not of question mark in the header. This variable works as an indicator if the header is a question or not.

In sequence we check the overlap of words between the header and the body, the amount of overlaps tend to indicate that they header and body are somehow correlated.

Evolving the previous idea we checked the cosine-similarity between the words in the header and the body, once again more similarities indicate a bigger chance that header and body are correlated.

Evolving the idea behind the previous two variables we created a variable which somehow combine the previous two. This variable was created based on selecting 5 randomised words from the headlines and 10 randomise words from the article body and run these words through Word2vec getting back 10 nearest words (cosine-similarity) for the body and 5 for the head.

It gives us 30 words for the header and 110 for the body. After that we check how many of the 30 words have correspondent into the 110 group. After finding how many they are we define the variable as the percentage with correspondence. That means if all 30 are found variable equals 100%. If none variable equals 0%.

The logic diagram goes as follows:



Finally we created a synthetic variable based on the function pairwise similarity from the [scikit-learn](http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer), which compares two TF-IDF phrases. If the results were similar to the 5 degree (equal until the 5 decimal) we consider them a match and give the variable value 1 if not we consider not a match and give it a value 0.

***Selecting variables***

We are going to analyze the relevance of each variable to the problem using four methods:

* Univarieted selection
* Recursive feature elimination
* Principal component analysis
* Feature importance

We are going to compare the five numeric variables regards their importance for our problem – Identify the correct instance.

Univarieted selection:

[3.626e+01 5.872e+01 2.919e+04 4.053e-01 5.973e+00]

That means :

* Question\_mark – 36,26
* Word\_overlap – 58,72
* Cosine-similarity – 29.190,00
* Cross - 0,40
* Match - 5,97
* Sent - 15,19

Recursive feature elimination

Num Features: 3

Selected Features: [ True True True False False False]

Feature Ranking: [1 1 1 2 4 3]

Recursive analysis choose the three first variables:

* Question\_mark – True
* Word\_overlap – True
* Cosine-similarity – True

Principal component analysis (PCA)

* Question\_mark – 0.099
* Word\_overlap – 0.184
* Cosine-similarity – 0.665

Feature importance

[0.002 0.115 0.794 0.088 0.002 0.002]

* Question\_mark – 0.002
* Word\_overlap – 0.115
* Cosine-similarity – 0.794
* Cross - 0.088
* Match - 0.002
* Sent - 0.002

Based on feature selection analysis we decided to select the following features "word\_overlap" and "cosine\_similarity" since they have higher importance.

***Approach /Analytical problem***

To analyze our data we are going to use two main strategies

1. Analyze the vectored texts without hyper-parameters (using all six methods)
2. Analyze the vectored texts with the hyper-parameters (using all six methods)

We are going to adopt the following strategy regards evaluating which analytical technique we are going to use. We are going to run 40 scenarios and will select the four ones which generate the best results regards the four parameters:

* Accuracy
* Precision
* Recall
* F1 score

Once we selected the four techniques (Including the technique and the scenario) we will “Ensamble” them into only one model. Then, we will apply this model to the test database and see the results.The models analyzed individually and the results were:



Once we run the code we have to evaluate the results, remembering we have to evaluate xx scenarios. Therefore we have to answer the simple question: How good is this scenario? So, evaluating the scenario is crucial to compare the alternatives and identify how good the strategy is in absolute and relative terms.

Just remembering the objective is to classify the articles regards the correlation between the article itself and its headline. The assumption is that if the article has no correlation with the header it probably is a fake news. Our training data has four possibilities regards the association between header and article:

* unrelated
* discuss
* agree
* disagree

Therefore the objective is to measure how assertive our code is classifying the articles correctly. We are going to analyze the scenarios using four parameters:

**Accuracy** - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same, this not our case, as we saw before 70% of the samples in our training data are “unrelated”. That means if we guess “unrelated” we will be right 70% of the cases.

Accuracy = TP+TN/TP+FP+FN+TN

**Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

**Recall (Sensitivity)** - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

**F1 score** - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially for us having an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

We selected the five scenarios highlighted in red. These five scenarios were combined using three Ensemble techniques:

***Ensembling models***

One way to make the models even better is to combine them. It is important because very often, we can manage to get a much better results when doing predictions from a combination of several models them from any one of them individually.

Here is important that we can adopt several possible strategies to combine the models: we may use the same technique and separate the reference database into small subsets and them combine the results of these individual models (bagging). Or we may use several techniques and combine the results of these models simultaneously into one final result (Stacking). Or we can combine the models two by two (Could be three by three) in sequence (Boosting).

We had basically five initial models:

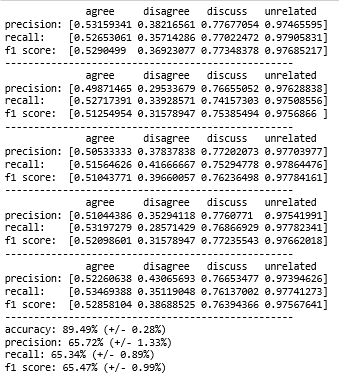


Initially we used decision tree and tested bagging with it

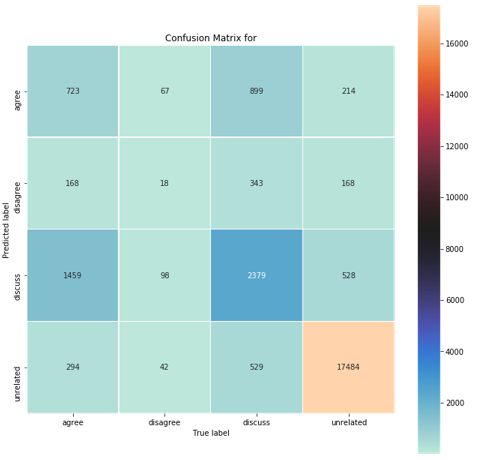
**Bagging**

We used Decision Tree with 10 bags then we ensemble these ten scenarios:

“bagging\_dt = BaggingClassifier(base\_estimator=pipeline\_dt, n\_estimators=10)”



The Confusion Matrix looks like:

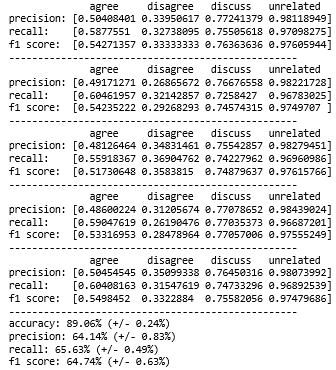


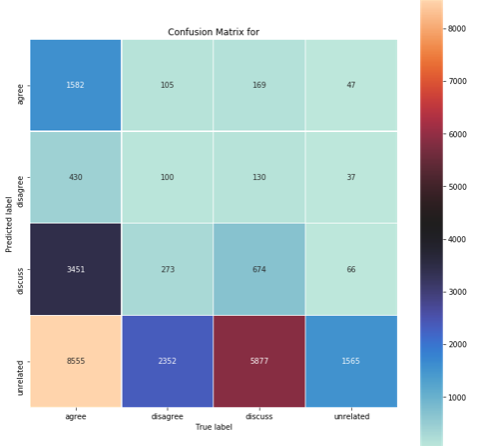
**Boosting**

We combined the models in the following sequence (boosting):



Summary:



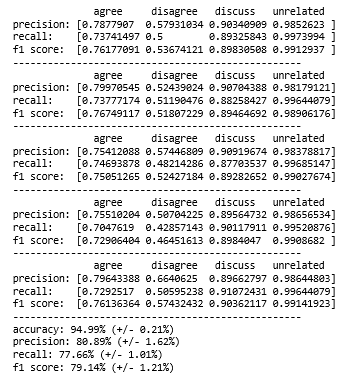


**Stacking**

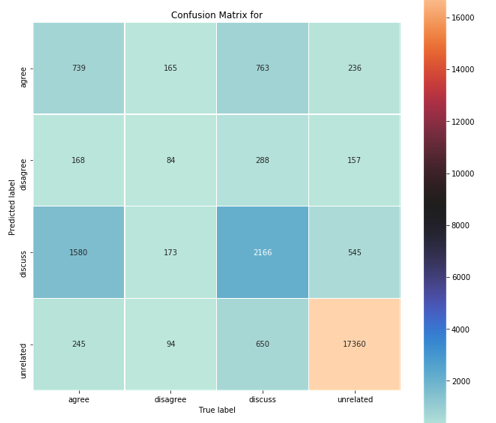
And finally we combined all five models in parallel (stacking)



Summary of the scenario:



This combined model gave as a confusion matrix as follows:

******

Note that the combined (Ensembled) is consistently better than any one of the five scenarios individually.

***Summary of findings /Evaluating the results***

Using the training data the two better models identified were the SVM and Ensemble 1:



However these results were obtained using our training data, when we apply the two models to our test data the ensabled model out-performs the SVM model:



Note that the outperform is not so much in terms of number of correct guesses but how well it performs when guessing minority classes. We can see that the Ensembled 1 model is approximately five times better in forecasting correctly the classes “Agree” and “Disagree*”.*

***Conclusion and Future Work***

*Fake news is still an open research topic. Further contributions are required, especially to deal automatically with the massive growth of information over the Web. Our work attempted to approach the stance detection of fake news using a simple model based on a combination of n-grams, word embeddings and lexical representation of cue words. These lexical cue words have been employed previously in the literature in rumors stance detection approaches. Although we used a simple feature set, we achieved similar results than the state of the art. This work is an initial step towards a further investigation of features to improve stance detection in fake news. As a future*

*work, we plan to focus on summarizing the articles in the dataset. As we mentioned in Section*

*3.2, the length ratio difference between the titles and the articles is large. Therefore, summarizing the articles may be a worthy attempt to improve the comparison between the two text fragments.*