Amarandei Alexandru, Mosor Andrei, Pelin Ioana, Smău Adrian

Team-Leader: *Cușmuliuc Ciprian Gabriel*

April 2021

CLEF2021 - CheckThat! Lab

Week 1 Report

Introduction

This team was assigned with the task of implementing an algorithm for detecting Fake News in Python. After doing the proper research and documentation, the team has decided to start implementing some simple approaches, based on the provided dataset.

Methodology

Overview

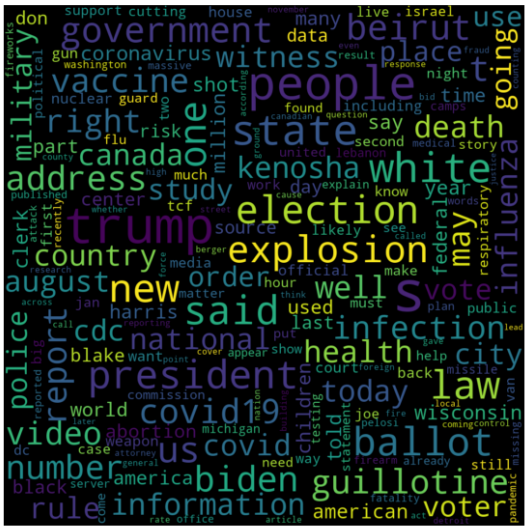
There were two main approaches: a word-frequency-oriented one and a numeric-encoding one.

For both of the approaches, after the data processing has been made, the DT (Decision Tree), KNN (K Nearest Neighbours), SVM (Support Vector Machine), and NB (Naive-Bayes) algorithms have been applied to the data, with 80% of the input being used for training and 20% for testing.

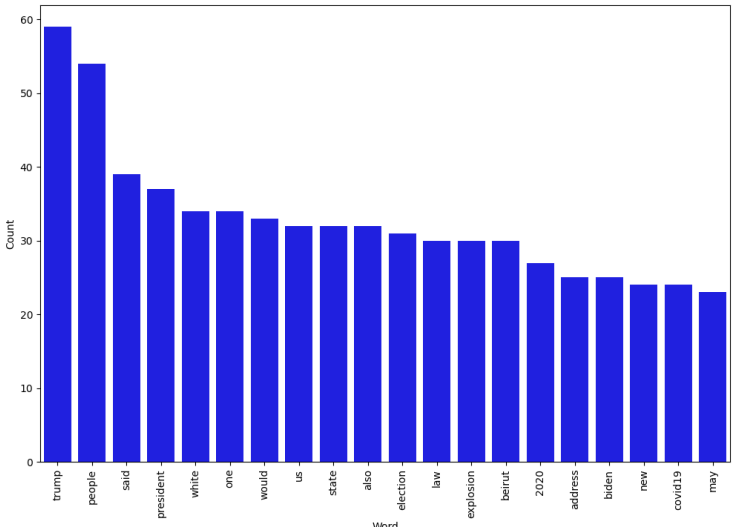
The performances of the algorithms have been tested using the F1-Score, Accuracy and Recall metrics, as well as the Confusion Matrix.

Word-Frequency Approach

The reasoning behind this approach was the fact that fake tweets often tend to contain certain words or group of words (such as “Trump” or “election” and “explosion”) and the analysis of the rate of appearance may lead to certain insights or tendencies.



Word Cloud View of the most frequent words in fake tweets



Bar Plot of the most frequent words in fake tweets

Before the Data Exploration has been made, the team had to prepare the data. During the Data Preparation stage, the text was transformed as follows:

- removal of the Title and ID columns from the CSV file received
- lowercase conversion of the whole document
- removal of punctuation signs
- removal of stop-words
- removal of dashes and underscores

~ Mention: the removal of the title and id columns, the case difference, as well as punctuation signs, were all deemed of lesser significance and impact during this first run, but are to be considered during follow-ups. The other removals will remain a part of the Data Preparation during future runs.

After finishing the Data Preparation and the Data Exploration stages, the team moved on to the application of the above mentioned algorithms using a

Pipeline composed of a Count Vectorizer, a TFIDF (Term-Frequency & Inverse Document Frequency) Transformer and the respective models.

The models were all trained using 80% of the input and tested using 20% of the input. A follow-up round used 90% / 10% as the train / test ratio.

The results are shown under the Results heading below and discussed under the Discussion heading right under it.

Word-Encoding Approach

The reasoning behind this approach was to check whether vectorizing the words using the TFIDF will lead to some insights that cannot be seen or thought of when working with character strings.


```
'newsmax': 2938, 'people': 3217, 'ground': 2029, 'germany': 1963, 'report': 3716, 'scytl': 3937, 'hosted':
2159, 'election': 1466, 'data': 1148, 'spain': 4142, 'raided': 3545, 'large': 2457, 'army': 300, 'force':
1843, 'server': 3994, 'seized': 3966, 'frankfurt': 1882, 'video': 4760, 'went': 4871, 'viral': 4775, 'say':
 3916, 'alleged': 184, 'raid': 3544, 'earlier': 1420, 'today': 4483, 'trending': 4560, 'fbi': 1747,
'sends': 3976, 'warning': 4827, 'advising': 114, 'buying': 623, 'making': 2626, 'misrepresenting': 2797,
'vaccine': 4729, 'card': 661, 'may': 2688, 'breaking': 569, 'law': 2472, 'later': 2465, 'posted': 3337,
'information': 2244, 'received': 3604, 'source': 4138, 'government': 2002, 'determined': 1280, 'dominion':
1373, 'involved': 2333, 'switching': 4358, 'vote': 4792, 'intelligence': 2283, 'community': 873, 'began':
446, 'search': 3940, 'discovered': 1328, 'order': 3076, 'get': 1964, 'access': 23, 'available': 373, 'use':
 4714, 'legal': 2499, 'manner': 2642, 'state': 4207, 'department': 1242, 'work': 4930, 'tandem': 4388,
'justice': 2392, 'request': 3729, 'cooperate': 1022, 'allowing': 191, 'seizure': 3967, 'appropriate': 282,
'document': 1366, 'required': 3733, 'affect': 120, 'kind': 2418, 'put': 3507, 'place': 3263, 'signed':
4059, 'appears': 269, 'also': 198, 'military': 2768, 'support': 4320, 'operation': 3057, 'lead': 2480,
'help': 2122, 'explain': 1676, 'esper': 1583, 'fired': 1799, 'miller': 2770, 'kash': 2399, 'patel': 3191,
'would': 4946, 'interfere': 2297, 'way': 4850, 'getting': 1965, 'ahold': 155, 'going': 1988, 'direct':
1309, 'evidence': 1620, 'instructed': 2271, 'stop': 4244, 'counting': 1052, 'discover': 1327, 'gave': 1943,
 'direction': 1311, 'algorithm': 178, 'started': 4204, 'cia': 774, 'completely': 892, 'excluded': 1640,
'think': 4446, 'democrat': 1227, 'tried': 4564, 'steal': 4219, 'yes': 4962, 'completing': 893, 'poll':
3304, 'entitles': 1551, 'gateway': 1942, 'pundit': 3496, 'news': 2936, 'update': 4693, 'free': 1890,
'charge': 732, 'opt': 3071, 'anytime': 255, 'agree': 148, 'privacy': 3401, 'policy': 3295, 'term': 4424,
```


Some of the words and their respective encodings.

The Data Preparation stage has been enriched with an extra step, compared with the one from the Word-Frequency Approach, that being the Lemmatization.

After removing the Title and ID columns from the data set and converting the texts to lowercase, each text has been tokenized into words. Following this step, a removal of stop words and non-alphabetical characters was made, the Data Preparation stage finishing afterwards with the Lemmatization.

The Data Exploration stage followed in a similar manner as the one before and the models used for testing were, again, DT, KNN, SVM and NB, the main difference being that Pipelines were not used during this run. The training / testing ratio used was also 80% / 20%, but a 90% / 10% was checked as well.

Results

Word-Frequency Approach

The results of this approach were not far from what the team's expectations were and yet, they were insightful.

80% / 20% Training to Testing Ratio:

```
Decision Tree
[[3 0 1]
 [0 2 0]
 [3 0 1]]
                precision    recall  f1-score   support

         FALSE       0.50      0.75      0.60         4
          TRUE       1.00      1.00      1.00         2
partially false       0.50      0.25      0.33         4

      accuracy                           0.60        10
     macro avg       0.67      0.67      0.64        10
  weighted avg       0.60      0.60      0.57        10
```

```
K-Nearest Neighbors
[[3 1 0]
 [0 2 0]
 [2 0 2]]
                precision    recall  f1-score   support

         FALSE       0.60      0.75      0.67         4
          TRUE       0.67      1.00      0.80         2
partially false       1.00      0.50      0.67         4

      accuracy                           0.70        10
     macro avg       0.76      0.75      0.71        10
  weighted avg       0.77      0.70      0.69        10
```

```
Support Vector Machine
[[4 0 0]
 [1 0 1]
 [4 0 0]]
                precision    recall  f1-score   support

         FALSE       0.44      1.00      0.62         4
          TRUE       0.00      0.00      0.00         2
partially false       0.00      0.00      0.00         4

      accuracy                           0.40        10
     macro avg       0.15      0.33      0.21        10
  weighted avg       0.18      0.40      0.25        10
```

```
Naive-Bayes
[[4 0 0]
 [1 0 1]
 [4 0 0]]
                precision    recall  f1-score   support

         FALSE       0.44      1.00      0.62         4
          TRUE       0.00      0.00      0.00         2
partially false       0.00      0.00      0.00         4

      accuracy                           0.40        10
     macro avg       0.15      0.33      0.21        10
  weighted avg       0.18      0.40      0.25        10
```

90% / 10% Training to Testing Ratio:

Decision Tree

```
[[2 0 1]
 [0 1 0]
 [1 0 0]]
```

	precision	recall	f1-score	support
FALSE	0.67	0.67	0.67	3
TRUE	1.00	1.00	1.00	1
partially false	0.00	0.00	0.00	1
accuracy			0.60	5
macro avg	0.56	0.56	0.56	5
weighted avg	0.60	0.60	0.60	5

K-Nearest Neighbors

```
[[3 0 0]
 [0 1 0]
 [0 0 1]]
```

	precision	recall	f1-score	support
FALSE	1.00	1.00	1.00	3
TRUE	1.00	1.00	1.00	1
partially false	1.00	1.00	1.00	1
accuracy			1.00	5
macro avg	1.00	1.00	1.00	5
weighted avg	1.00	1.00	1.00	5

Support Vector Machine

```
[[3 0 0]
 [0 0 1]
 [1 0 0]]
```

	precision	recall	f1-score	support
FALSE	0.75	1.00	0.86	3
TRUE	0.00	0.00	0.00	1
partially false	0.00	0.00	0.00	1
accuracy			0.60	5
macro avg	0.25	0.33	0.29	5
weighted avg	0.45	0.60	0.51	5

Naive-Bayes

```
[[3 0 0]
 [0 0 1]
 [1 0 0]]
```

	precision	recall	f1-score	support
FALSE	0.75	1.00	0.86	3
TRUE	0.00	0.00	0.00	1
partially false	0.00	0.00	0.00	1
accuracy			0.60	5
macro avg	0.25	0.33	0.29	5
weighted avg	0.45	0.60	0.51	5

Word-Encoding Approach

Taking into consideration this approach's purpose, the results were helpful for deciding that disregarding the words and their meaning is certainly not efficient.

80% / 20% Training to Testing Ratio:

Decision Tree

```
[[3 0 4]
 [1 0 2]
 [0 0 0]]
```

	precision	recall	f1-score	support
0	0.75	0.43	0.55	7
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	0
accuracy			0.30	10
macro avg	0.25	0.14	0.18	10
weighted avg	0.53	0.30	0.38	10

K-Nearest Neighbors

```
[[4 1 2]
 [0 2 1]
 [0 0 0]]
```

	precision	recall	f1-score	support
0	1.00	0.57	0.73	7
1	0.67	0.67	0.67	3
2	0.00	0.00	0.00	0
accuracy			0.60	10
macro avg	0.56	0.41	0.46	10
weighted avg	0.90	0.60	0.71	10

Support Vector Machine

```
[[1 0 6]
 [0 2 1]
 [0 0 0]]
```

	precision	recall	f1-score	support
0	1.00	0.14	0.25	7
1	1.00	0.67	0.80	3
2	0.00	0.00	0.00	0
accuracy			0.30	10
macro avg	0.67	0.27	0.35	10
weighted avg	1.00	0.30	0.42	10

Naive-Bayes

```
[[1 0 6]
 [0 0 3]
 [0 0 0]]
```

	precision	recall	f1-score	support
0	1.00	0.14	0.25	7
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	0
accuracy			0.10	10
macro avg	0.33	0.05	0.08	10
weighted avg	0.70	0.10	0.17	10

90% / 10% Training to Testing Ratio:

```
Decision Tree
[[1 1 1]
 [2 0 0]
 [0 0 0]]
              precision    recall  f1-score   support

           0       0.33      0.33      0.33         3
           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         0

    accuracy                           0.20         5
   macro avg       0.11      0.11      0.11         5
weighted avg       0.20      0.20      0.20         5
```

```
K-Nearest Neighbors
[[1 1 1]
 [0 1 1]
 [0 0 0]]
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.50      0.50      0.50         2
           2       0.00      0.00      0.00         0

    accuracy                           0.40         5
   macro avg       0.50      0.28      0.33         5
weighted avg       0.80      0.40      0.50         5
```

```
Support Vector Machine
[[3 0]
 [1 1]]
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       1.00      0.50      0.67         2

    accuracy                           0.80         5
   macro avg       0.88      0.75      0.76         5
weighted avg       0.85      0.80      0.78         5
```

```
Naive-Bayes
[[3 0 0]
 [1 0 1]
 [0 0 0]]
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00         0

    accuracy                           0.60         5
   macro avg       0.25      0.33      0.29         5
weighted avg       0.45      0.60      0.51         5
```

Discussion

Word-Frequency Approach

From an objective standpoint, the KNN algorithm performed the best out of the four that have been chosen. Furthermore, this result is also backed by an intuitive reasoning, as similar tweets tend to be in the same category (True, False or Partially False).

The DT and NB did not outperform or underperform the expectations of the team, however, the SVM's low accuracy led the team to conclude that in the case of three separate categories (True, False and Partially False, instead of just the first two) the algorithm can't make an accurate enough linear separation of the three sets, two disjoint sets being the optimal number.

The small input size also made the team question whether or not the algorithms could have performed better with more training data.

Modifying the train-to-test ratio to 90% / 10% lead to better results, however, there are multiple possible causes at play: better training, less tests to be failed, overfitting.

Better Training:

The obvious line of reasoning is that along with a bigger sample of tweets, there will be an increase in the accuracy of the detection and classification. The team believes that this, along with the reason immediately below, probably best describe the real story behind the results.

Less tests to be failed:

The training and testing sample sizes being proportional, it is only logical that a bigger training sample will lead to a smaller testing sample, which, in turn, reduces the number of possible failures. While it is certain that most of the time, more training translates to better results, this may also imply getting poor insights from the testing results or even misleading ones, due to overfitting.

Overfitting:

Overfitting happens when a model learns the detail and noise in the training data to such an extent that it negatively impacts the performance of the model on new data. Too many similarities occurring between individual samples may also result in a decreased generalisation ability, which represents the main concern of the team on this subject.

For instance, training the model on too many common-subject tweets (e.g. politics or COVID-19 related) may negatively impact the performance on tweets on other subjects (e.g. nutrition myths, fabricated history) or even on news articles. The team has taken into consideration these facts and came up with suggestions (e.g. Decision Tree Pruning) which will be implemented in future runs.

Word-Encoding Approach

As speculated, the results were less than satisfactory, the only model with decent performance being the KNN. The logical conclusion that the team drew was to discontinue researching in this direction.

The training-to-testing ratio adjustment further enforced the idea that results could be discarded, as the NB model's performance has jumped from an accuracy of 10% to 60%, whereas the DT's and KNN's performances dropped.

Future Approaches

Data Preparation

The ideas conceived by the team will be presented decreasingly by the order of importance and relevance.

Titles:

The team decided that tweet titles are also relevant in the current context, as such, they are not to be removed as of now and a special approach (different from the one designed for the body of a post) will be implemented to make the title analysis as fruitful as possible.

Emojis:

Emojis have become a standard part of tweets or chats in the last few years and are now capable of conferring complex meanings and emotions, which may alter the initial sense of the post. They must be taken into consideration as soon and as well as possible in order to get a complete image of the text nuances.

Lemmatization:

Although the approach in which lemmatization was used will not be improved on, this step will remain in the text transformation toolbox as it facilitates the grouping of words that have the same lexical origin.

Models

This first run's main purpose was the accommodation of the members with the new technologies and libraries, as well as designing a few basic algorithms to get a grasp on the task that the team is facing.

The goal was successfully achieved and, after the deep analysis of the results, the team is ready to move on and experiment with BERT (Bidirectional Encoder Representations from Transformers) and NLP (Natural Language Processing).

The first is a relatively-recent innovation that translates to a language model that is bidirectionally trained (although the Transformer encoder reads all the word sequences at once, making it more non-directional).

The latter is a necessary means of taking topics of discussion and emotions into consideration.