Coursework 1

COMP3222 Machine Learning Technologies

gt2u19@soton.ac.uk

Introduction and Data Analysis

Currently, the majority of social media platforms do not have the tools to classify propagating fake textual messages concerning viral events effectively. The task at this time is to assist professionals in discerning the credibility of user-generated content. This will be done by implementing a machine learning algorithm with the provided datasets and classifying twitter messages as "real" or "fake". This assignment is inspired by the "MediaEval 2015" multimedia verification task.
The datasets provided come in the format of tab-separated text files. Each line represents a different message, with the following information

- 'tweetId' - the internal identifier of the tweet
- 'tweetText' - the message which was posted
- 'userId' - the internal identifier of the poster
- 'imageId(s)' - internal identifiers for the images attached
- 'username' - the public username of the poster

- 'timestamp' - the exact time of posting
- 'label' - the ground truth label of the message(either 'real', 'fake' pr 'humor')

After examining the training data set, a couple of things stand out:

- Only the 'tweetText' and 'label' columns containt relevant information to the task.
- The ratio of "real" : "humor" : "fake" is around 1 : 1 : 2. Since we are to treat the humor label as fake, the ratio "real" : "fake" is approximately 1:3.
- The number of entries is 14484, depicting 11 unique events.
- The average number of words is around 9.8.
- The average length of a word is 5 characters.
- The average number of stop words in a "tweet" is 2.4.
- In nearly 24% of "tweets" non-ascii characters appear, the most common ones being Spanish, Arabic, Cyrillic and Chinese.
- When looking at random samples of the most uncommon words, a lot of them appear to be URLs.
- The most common words appear to be English and Spanish stop words and 'hash-tags' of over-represented events (like "#Sandy").
- There appear to be a lot of punctuation symbols and 'emoji' characters.
- There are end of line characters "/n" present.
- Grammatical errors and spelling mistakes also are quite common within the data set.

Overall, there seems to be a slight selection bias in favor of English 'tweets', written by English-speaking users. Furthermore, the Twitter user-base represents a narrow demographic subset(20-40). However, these biases are negligible and can not be mitigated within a Twitter data set and should ,therefore, be accepted.

Algorithm Design

Pre-processing

For this part of the work, the pandas/nltk libraries for pre-processing and pyplot for visualisation were chosen. The approach was inspired by Nicollas Oliveira[3].The following steps were taken in the given order:

1. Import the training set using the pandas read_csv() function, by specifying a tab separator.
2. Only the 'tweetText' and 'label' columns containt relevant information to the task. Therefore, we drop all other columns(e.g 'Timestamp', 'Username', etc.).
3. Change all characters in the 'TweetText' column to lower-case. This was tested practically, and proved beneficial to the algorithm score.
4. Remove all "/n" "/r" characters using a regex. These do not provide any learning value to the algorithm and should be removed.
5. Remove all URLs from the text column using a regex(note that, there were 33 URLs with a different format, such as "http:// 'urlstring'" which are also caught by the regex). From the data analysis, it was apparent there are a lot of unique URLs in the data set. Since the special characters are to be removed, the contents of the URLs will later become hard to capture and should be removed during the current stage.
6. Next, all non-ascii characters were removed, using the internal function encode_ascii(). This decision was made to avoid word translation as it may be too computationally expensive, and may potentially even lower the overall score.
7. Because of the nature of "Twitter", a lot of the messages incorporate "hashtags" in the format "#_" which represent keywords and encapsulate part of the message. Precisely for that purpose, during this stage, all of the unique "hashtags" are saved in a local variable.
8. Remove punctuation and empty spaces, then remove all stop words. This is done to reduce the number of characters, leaving us with a clean set of words(the most useful ones).
9. After removing all stop words, the text is lemmatized(using the nltk Lemmatizer). This option was chosen over alternatives, such as the nltk stemmer. This is due to personal testing, during which the lemmatizer outperformed the stemmer, and was more practical in the following step.
10. Removed non-English words(except "hashtags"). This was done to test my hypothesis, that using only the English would increase overall accuracy(because of tools

like the lemmatizer). Another advantage of keeping the English "hashtags" but removing the non-English text is that most of the "hashtags" are in English and are essentially keywords for the message.

11. Lastly, all 'humor' labels are changed to 'fake'. Rather than deleting the 'humor' labels, keeping them in the data set was incredibly effective(after testing both approaches, the latter configuration achieved over 20% higher F1 score).

Feature Extraction

TF-IDF

Feature extraction was done using the TF-IDF vectorizer. "TF" refers to the term "Term Frequency", which essentially represents the ratio of specific word appearances, divided by the number of words in the sentence. The term "IDF" is short for "Inverse Document Frequency", which provides a ratio between the total number of rows to the number of rows where each specific word appears. "TF-IDF" is simply the values of the previous two terms multiplied.

N-grams

Another improvement to the accuracy is implement "N-grams". They are pairs or triples(or n-tuples) of words frequently used together. Those are then forwarded to the "TF-IDF" to find the discriminating value for each n-gram. For the current configuration, "unigrams" and "bigrams" yielded the most optimal results. During testing, "trigrams" appeared to reduce the score and increase computing resources, one potential cause for this might be the short word length of each tweet after the pre-processing. This would lead the algorithm to focus on very specific features, rather than the general cases.

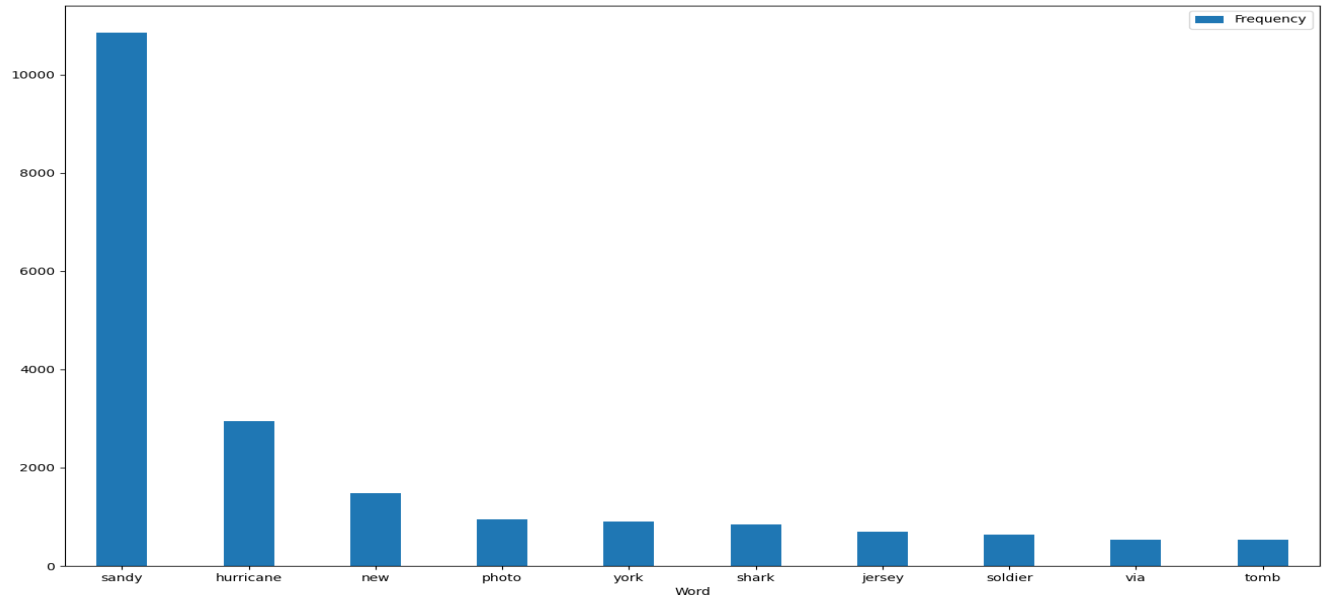

Figure 1: Top 10 features by number of occurrences

Feature Analysis

As shown in Fig.1, the top features appear to be very topic-specific, and very common. Such features only decrease the accuracy and should therefore be removed, in order to avoid over-fitting. Furthermore, when looking at the data set, a large number of features have only a single occurrence. This indicates that a limit should be implemented to both minimal and maximum occurrences. Through testing, the minimum occurrences ("min_df") should be 0.01%(to filter the singletons) and maximum number of appearances("max_df") should be about 10%(after testing, this was enough to avert over-fitting and yet discard the most problematic features).

Algorithm

The Random Forest Classifier is a meta-estimator which works by fitting several decision trees on fragments of data and averages in order to control over-fitting and efficiency. This algorithm provides both high stability and accuracy, which is why it was chosen initially. The number of classifiers(or estimators) chosen for this problem was 10. Generally, as the number of decision trees increases, so does the accuracy, however, the computational weight increases exponentially as well. The number of choice has not been tweaked too much to avoid over-fitting to specific cases, rather the aim was to produce an algorithm which would perform well with different, contrasting data sets.

Evaluation

After the implementation, numerous changes were made to the pre-processing and the specific values of the algorithms, which will be briefly summarized in the following subsection.

Iterative Improvements

1. The 'humor' label could either be removed or treated as 'fake'. Empirical testing showed immense loss of accuracy when those labels were deleted. This is due to the high volume of 'humor' labels which account for about $1/4$ of all data.

2. All non English words were removed from the 'tweet' text column by using an nltk dictionary of english words. This yielded a substantial improvement to the number of features (>10000 -> 6300) and also to the F1 'Micro' score(77% -> 88%). However, after experimentation, this number of features may have led to some over-fitting. This warranted a limit to the maximum features(set to 2000) to avoid such a risk. This led to an accuracy of 83%.

3. In regards to the previous step, the English dictionary was extended to include all 'hashtags' from the 'text' column. This increased the number of features by about 500, and improved the F1 score by approximately 2 points(to 0.856).

4. After carefully examining samples of the resulting training set, some irregularities were alerting(e.g. "e", "en", "de"). Words with 2 letters or less appeared to go through the English dictionary "sieve". To counter this, all words of length 2 or less were removed. Surprisingly, this improved the accuracy by more than 1% to 0.866.

5. The last improvement made to the pre-processing was made after a data visualisation, where several rows of the training set were left with no features. This was likely caused by removing all ASCII incompatible symbols and keeping only the English words. The empty lines were removed from the training data, which resulted a brought the F1 score up to 0.870

Practical Results

The following section outlines the performance of the best configuration for each algorithm. After testing in practice, the maximum/minimum number of features in TF-IDF were set to generic 0.1 and 0.0001 to avoid over-fitting to specific values and the number of features were set to top 2000. Regarding the Random Forest Classifier, when using the criterion 'entropy', the results were more accurate than when using the default 'gini' index. The K-Nearest Neighbors' configuration was done with 10 neighbors and the Support Vector Machine and Multinomial Naive Bayes were left to default setting.

Algorithm	F1	TP rate	TN rate	Training Time	Testing Time
LinearSVM	0.870	0.882	0.840	109s	37s
K-Nearest Neighbors	0.851	0.843	0.878	20.1s	7.3s
Multinomial Naive Bayes	0.845	0.887	0.758	20.7s	5.5s
Random Forest	0.823	0.862	0.736	25.1s	5.7s

Table 1: Algorithm Results table

In the above table, the F1 metric represents the 'Micro F1' score(between 0 and 1). The TP-"True Positive"(correctly identified 'real') and TN-"True Negative"(correctly identified 'fake' labels) rates show the algorithm's ability to handle specific label types. The training and testing times show the time to train the data set and predict testing set in seconds, respectively. After researching[1], those metrics were chosen as they provide great insight into each algorithms performance.

Conclusion

The methodology used yielded very promising results. Every step of the design was made with the intention of creating an algorithm, which would perform well practically. This led to some decisions which reduced the F1 accuracy slightly and yet the number of features decreased by nearly 5 times(to 2000). The chosen approach of keeping only words from the English dictionary showed a great accuracy gain across all algorithms. After considering all results, the Linear Support Vector Machine achieves the best accuracy overall. However, the Multinomial Naive Bayes shows a substantial improvement in testing/training time. For the current assignment, the LinearSVM is clearly the best algorithm in terms of accuracy. However, in an environment where a low prediction time is vital, the 3% accuracy loss may be overlooked in favor of a 85% prediction speed improvement.

Future Improvements

Firstly, one problem which was unaddressed with my algorithm design, is the problem of intentional/unintentional spelling errors. The TextBlob[2] library provides a spelling-correcter, its only downside is the computational time required. Another potential improvement is the addition of language translation with the Google Translate Python library. The most practical improvement, however, is to improve the English language filter. During the evaluation, it became clear that some Spanish words would bypass the English detection. The exclusion of those words could possibly improve the accuracy by up to 2%.

Learning Outcomes

During the research and implementation of this work, several insights were gained:

- The highest F1 score is not the goal for this machine learning algorithm. In fact several times during implementation, the F1 metric was not even considered when making generalised improvements.
- With a large amount of features(e.g. >10000), over-fitting is inevitable[4]. To prevent this from happening, filters should be implemented to extract only the best features.
- Pre-processing the data correctly greatly reduces the risk of over-fitting to a specific data set.
- Data visualisation techniques(such as the ones, presented in the lectures) provide very valuable insights as to how the algorithm works. Identifying the flaws of an algorithm becomes considerably easier.

References

[1] Aditya Mishra. "Save Your Data with These Empowering Password Statistics". In: *Venture Capital* (February 24, 2018). URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[2] *Natural Language Processing for Beginners: Using TextBlob*. February 11, 2018. URL: https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/.

[3] Nicollas R. de Oliveira et al. "Identifying Fake News on Social Networks Based on Natural Language Processing: Trends and Challenges". In: *Information* 12.1 (2021). ISSN: 2078-2489. DOI: 10.3390/info12010038. URL: https://www.mdpi.com/2078-2489/12/1/38.

[4] Shaeke Salman and Xiuwen Liu. *Overfitting Mechanism and Avoidance in Deep Neural Networks*. 2019. arXiv: 1901.06566 [cs.LG].