HIO Language Processing 2, Spring 2020

Exam submission – Group assignment

Group Participants:

Ting-Yu Kuo – kvq941
Mengran Chen – hcq596
Lena Burkel – gsw443

Supervisors:

Manex Aguirrezabal Zabaleta
Jürgen Wedekind

June 9th, 2020

Table of Contents

Profiling fake news spreader from twitter

Ting-Yu Kuo	Lena Burkel	Mengran Chen
kvq941	gsw443	hcq596

Abstract

Nowadays, huge amounts of news posts on social media are making it increasingly important to identify fake news from real news, as billions of people experience news and events from social media platforms like twitter. We have used various supervised learning methods like AdaBoost, Logistic Regression, Random forest, and Support Vector Machines (SVM) on the data to differentiate fake news spreaders from non fake news spreaders. For our machine learning models, we have artificially generated 22 features based on our linguistic and common sense knowledge, and we also have used a long short-term memory-based (LSTM) neural network to extract more features. With only artificially generated features, we achieved an accuracy of 0.66 using linear SVM, and a 0.63 accuracy using Random Forest classifier. However, after including the features extracted by LSTM-based network, we achieved an accuracy of 1.0 in all of our machine learning models.

1. Introduction

Since the establishment of the internet, people have started implementing the world wide web (www) not only in their work lives but also for private activities. The higher interaction with that technology includes social media, too. Its opportunity to stay connected all day long with friends makes it an attractive new way of communicating (see Shu et. al 2017:22). Furthermore, we now have cheaper and easier access to connect with persons from all over the world, no matter which times zones, countries and continent which in addition boosts the use of such communication in companies where interaction is continuously more moved to email than via phone – and which is why many argue that working became more efficient. This does not only apply for industrial or trading companies that can now stay connected with their branches in all the corners of the

world. It also accounts for every other aspect of both personal lives and jobs. The increased use of the internet allows everyone with internet access to participate in any online activity, giving them the opportunity to freely express their opinions and feelings on everything they wish to comment on. This new way of online interaction especially affects politics: according to Shu et al (2017), "social media now outperforms television as the major news source" (p.1). However, the content of such social media articles and posts, do not always represent the truth nor state facts. Fake news are articles or posts that intentionally give false information. In consequence, the amount of so-called *fake news* is increasing (see Pérez-Rosas et al 2017: 1). Pérez-Rosas et al (2017) refer to a report by Facebook stating that 50% of their posts and news were fake news - only 20% were reliable websites. When at the same time, 62% of US adults mainly inform themselves via social media (Pérez-Rosas et al 2017: 1, Shu et al 2017: 22), the risk of negative impacts of fake news can be critical.
Motivations for that can, among others, include political intentions, such as manipulating elections (see Shu et al 2017: 22, Tschiatschek et al 2018: 517). A prominent recent example for this is the dispute on Russia's manipulation (see Tschiatschek et al 2018: 517) of social media activities during the presidential elections in the US in 2017. That type of manipulation of political dynamics can have great impacts on the way we do and perceive politics (e.g. Ruchansky et al 2017: 797), such as lack of trust and authenticity and a change in belief systems when readers take fake news for true.

In order to ensure the well-being of political systems, it is crucial to identify such fake news and their sources. With the increasing development of technology, we permanently find new ways of detecting these news. However, as the ability of detecting fake news increases, so does the quality of fake news, meaning that the way fake news is written seems always more authentic, making it difficult to identify. In their study "News Detection Through Multi-Perspective Speaker Profiles" from 2017 Long et al state that the humans they studied could only identify 50-63% of the fake news as such (p.1). That is one of the reasons that new websites for fact-checking, such as "Factcheck.org" or "politiFact" have arised that check and analyse fake news (see Tschiatschek et al 2018: 517). Apart from that, also computer science develops new approaches for such detection.

As fake news presents a “new” truth, the content is important but also the way that truth is presented and what kind of language is used for doing so. That can include a reinterpretation of true facts which leads to a seeming truth as it is based on facts (see Shu et al 2017: 23). Such aspects need new effective ways of detection which is what this paper will be about.

In this paper, we aimed at training machine learning models such as Random Forest classifier, Support Vector Machines to be as much accurate as possible in differentiating fake news spreaders from non fake news spreaders. For that, we extracted linguistic features (e.g. lexical diversity, POS tagging features, etc.) and user behaviours such as hashtag and retweet frequencies from our dataset, which contained 300 English twitter users and 300 Spanish twitter users with 100 tweets per user, half of whom were fake news spreaders and another half were not. We also implemented a long short-term memory (LSTM) based neural network and directly trained on corpus. The trained weights of the last hidden layer in the LSTM-based network were extracted as features to fit on our machine learning models.

As fake news highly depends on click baiting, meaning that the news are spreading very quickly based on eye catchers or specific linguistic cues that attract attention, we expect - from a linguistic point of view - to find a great use of emotional words, e.g. to express intense emotions and exaggerations, accompanied by a possibly higher use of exclamation marks to stress the emotional value of the message. Furthermore, we’d assume less complex and shorter sentences and opinionated language.
Regarding the profile and the quality of the news, we expect to find that the source of fake news may be much less diverse than that of real news, so that the writing style of fake contents may tend to be more uniform than real contents, which may lead to a lower lexical diversity in fake news. Also, we will analyse retweet frequency and cross-post URL frequency to determine if there might be fake accounts that specialised in distributing fake news.

2. Dataset description

We used the dataset of the shared task “Profiling Fake News Spreaders on Twitter” at PAN 2020, which is a platform for a series of scientific events and shared tasks on digital text

forensics and stylometry. The dataset consists in two folders corresponding respectively to two languages: English and Spanish. Both folders contain 300 XML files, and each of them represents a Twitter user ID with 100 tweets. As we mainly focused on English data, we mainly dealt with the 300 English speaking users, who posted 300*100 tweets in total. We can see the short texts of all the tweets, including tags showing user behaviours such as RT (retweet), #USER# (mentioning other users), #URL# (post other URLs), and #HASHTAG# (hashtags). However, there is no access to other relevant account information, such as post time, number of followed or following people, etc. All the user IDs have been assigned with a ground truth label indicating whether they are fake news spreaders (1 for fake news spreader, 0 for non fake news spreader). Among them, half are fake news spreaders, and the other half are not. After the exploration of English data, we also tested the model on Spanish dataset.

3. Methods

3.1 Overview

We presented an ensembled model that combines a LSTM-based neural network and non-neural network classification algorithms. There are four different stages in the whole approach. Firstly, we clean the raw corpus by removing Twitter tags and generating 22 features based on our linguistic knowledge and hypotheses to fake news spreaders (referred as "artificially engineered features"). Secondly, we trained the LSTM network on the tweets content. Thirdly, the last layer of the LSTM network was extracted as new features. At last, all features were combined together and were trained on several different algorithms such as logistic regression, Random forest, etc.

3.2 Preprocessing and artificial feature engineering

We generated individual features for the tweets of both fake news spreaders and non fake news spreaders. For extraction of most features, the data needs to be subjected to certain preprocessing steps at first. Thus, we developed a clean function using the regex package in Python to remove punctuations, URLs, hashtags, mention tags and retweet tags, and we also used spaCy and nltk packages to perform tokenization and/or lemmatization. The features can be classified as lexical

features, semantic features, syntactic features, sentiment features and user behaviours. Detailed description of the features can be found in the following subsections.

3.2.1 Lexical features

For lexical features, we extracted three: a) word count, b) stopword count and c) lexical diversity.

Lexical diversity is the range of different words used in a text, with a greater range indicating a higher diversity (McCarthy and Jarvis, 2010). We first tokenized the tweet texts with spaCy package, and then measured lexical diversity using the Python project lexical-diversity 0.1.1. It basically calculates the type/token ratio, which is the vocabulary size divided by the number of tokens.

3.2.2 Semantic features

We generated five semantic features: a) count of numbers, b) count of named entities, c) count of question marks, d) count of exclamation marks, and e) count of manually generated keywords. We used the spaCy package to recognize named entities in the tweets, which are “real-world objects” with an assigned name (e.g. a person, a country, a book title, etc.), and we counted the number of them for each user. We also counted the number of the punctuations “?” and “!” , and also cardinals in the tweets of each user. At last, we manually generated a list of keywords to match them against the tweet texts and count the number of all them in the tweets of each user. We assumed that a large proportion of fake news would be related to political or celebrity issues, so we selected some frequently used words in these fields. Here is the list of the keywords: ["trump", "obama", "biden", "clinton", "us", "iran", "rubio", "gop", "islam", "islamic", "u", "top", "kim", "bush", "visa"]. We included “u” here because the word “U.S.” would become “u” after preprocessing, and we also included the word “top” as a lot of fake news have clickbait titles similar to “Top 10 places in the word”.

3.2.3 Syntactic features

The five syntactic features we included are: a) adjective frequency, b) verb frequency, c) noun frequency, d) adverb frequency, and e) pronoun frequency.

In order to extract them, we firstly applied the spaCy package to perform part-of-speech tagging (POS tagging) for each tweet. Then we counted respectively the number of POS tags referring to adjectives (ADJ), verbs (VERB), nouns (NOUN), adverbs (ADV) and pronouns (PRON) used by each user.

3.2.4 Sentiment features

With sentiment analysis, we extracted four sentiment features: a) mean of sentiment values, b) standard deviation of sentiment values, c) mean of subjectivity values, and d) standard deviation of subjectivity values.

We applied the TextBlob package for sentiment analysis. Internally, this package has a corpus consisting of negative/positive words and objective/subjective words. The text data is matched against the corpus, based on which the polarity score and subjectivity score are assigned. The context of each word is taken into consideration while assigning the scores to a text. The polarity is a float within the range [-1.0, 1.0], where the score for positive words are higher than 0.0, and the score for negative words are below 0.0. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective, and assessments is a list of polarity and subjectivity scores for the assessed tokens (TextBlob 0.16.0 documentation). With polarity and subjectivity scores assigned to each tweet, we then calculated the average value and standard deviation of these scores for each user.

3.2.5 User behaviours:

There were five user behaviours we observed: a) cross post, represented by the number of tweets containing #URL# tag of each user; b) duplicated cross post, i.e. number of the tag #URL# in the tweets of each user; c) retweet count, d) mention count, and e) hashtag count, which are numbers of the tags “RT”, #USER# and #HASHTAG# in the tweets of each user, respectively.

3.2.6 Principal component analysis

The principal component analysis (PCA) method was performed on the features. PCA is a dimension reduction method that is commonly used to visualize data in high dimension. Here we used it to visualize the relationship between our features.

3.3 Modeling

We applied an ensemble model in the current project (see *Fig. 1*). For the first part, a LSTM-based neural network was trained on tweets data. For the second part, we extracted the trained weights inside the LSTM-based network and appended it into our artificially engineered feature. The key concept was trying to include all available data. Our artificially engineered features merely contained the metadata of corpus, which was generated based on our linguistic knowledge and hypotheses to the fake news spreader. However, those features did not include the first-hand content of tweets. Therefore, the core idea of model design is to construct a model in which able to evaluate the linguistic metadata and the corpus simultaneously.

In order to train the LSTM-based network, the tweets corpus was flattened, preprocessed (to exclude Twitter tags and punctuations) and tokenized (for more details, see section 3.2 *preprocessing and artificial feature engineering*). The original binary label was transformed into one-hot encoding data, i.e. [1, 0] and [0, 1]. The LSTM-based network contained 100 LSTM units, then connected to a fully-connected (dense) layer with 100 units and another dense layer with 20 units. Here, we trained and validated the LSTM-based network only on the training set. The validation was set to 10%. Therefore, only 60.03% of the raw data was used for LSTM-based network training (66.7%* 90% = 60.03%). The training of this network reached its best performance on its validation data (i.e. 66.7% * 10% = 6.7% of the full dataset) for 1 epoch. The accuracy on validation data was around 0.72. However, the accuracy here is not our concern. We considered the LSTM-based neural network as a "weak learner" that played its role in learning to identify the fake news spreader from their tweets directly, instead of the metadata of their tweets (as our artificially engineered features). As long as the accuracy of the LSTM-based network is above random guess (accuracy > 50%), the training result was considered acceptable.

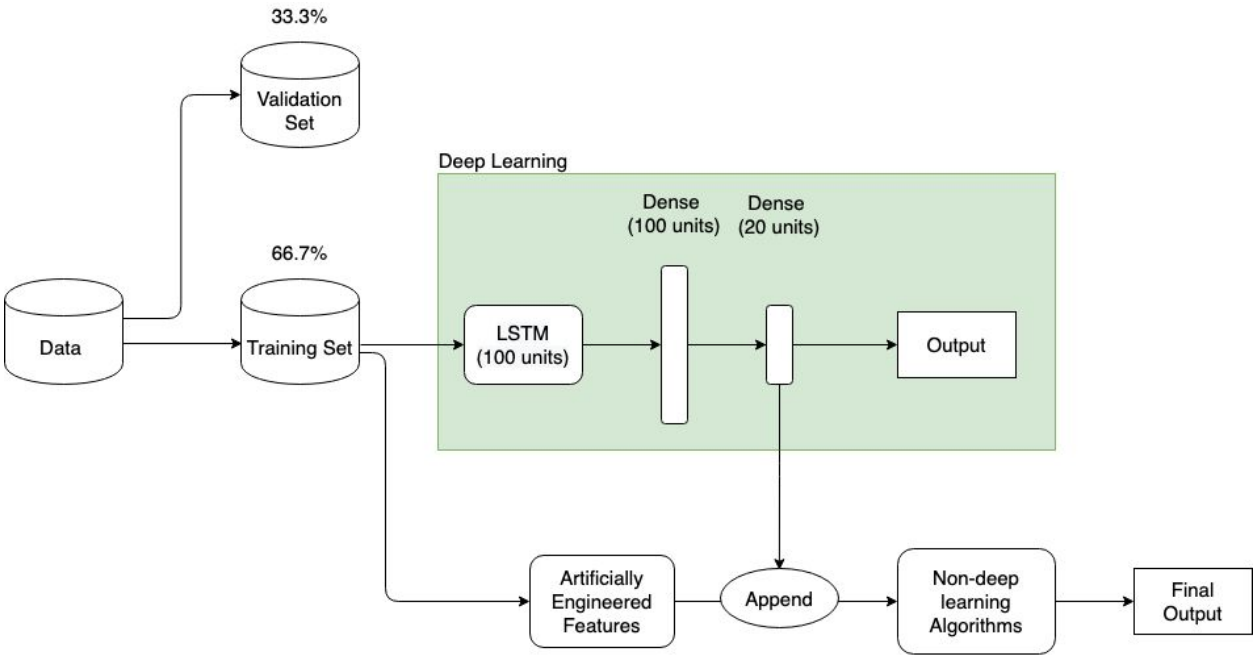

Figure 1. Model structure. This graph demonstrates the model structure. The region with green background is a LSTM-based network. The last dense layer was extracted and appended to the dataframe with artificially engineered features.

When the LSTM-based network was trained. The trained weights of its last layer were extracted and appended to the previous artificially engineered feature dataframe. This method was inspired by the previous research on the visualization of layers in convolutional neural networks. (Erhan 2009) and pre-trained word embeddings such as BERT(Devlin & Toutanova 2018) and word2vec (Mikolov and Dean 2013). Consider a binary classification task as an example. Each artificial neuron inside of a deep neural network could split the whole task and work on only a small proportion of it. As the information flows from the input to the output layer, the last layer should contain all processed information of the tweets. A common implementation of classification task on LSTM is to connect a multi-unit dense layer as the output layer, where the numbers of the units should align with the number of classes. In our case, we extract the dense layer before the output layer. The shape of that was (20, 2), where 20 means the number of neurons, and 2 means **True** and **Fake**, in alignment with the classifying target. We appended the trained weights extracted from the last hidden layer to the feature dataframe. Therefore, 20 new columns were appended, and each of them contained only 2 different values. For LSTM training, we applied the keras package. The codebase was referenced to (12).

Next, we trained these features on several algorithms and compared the results. Here, we tried 4 algorithms that are commonly used for classification tasks, including: AdaBoost, Random Forest, Logistic regression, and different kernels of SVM. (for more detail comparison, see *Results* section). For model training, we applied the Scikit-learn package on Python.

4. Results

4.1 Artificially engineered features

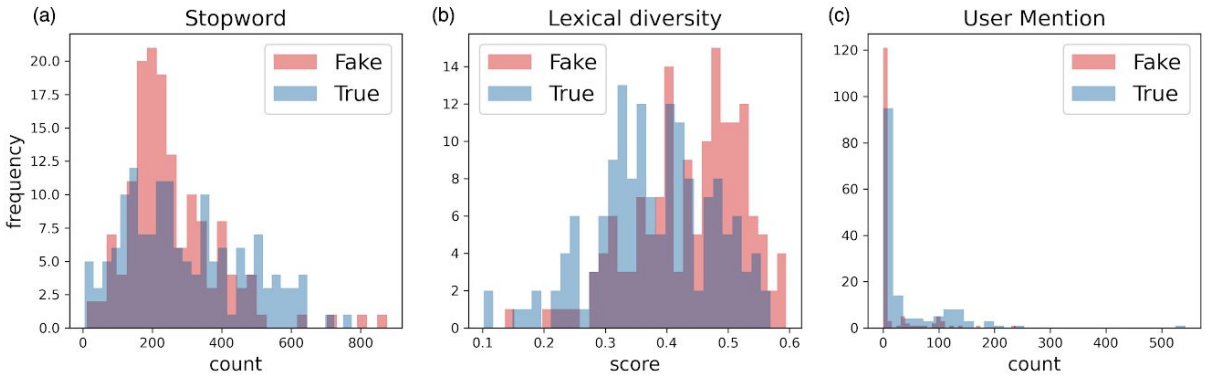

Figure 2. Frequency distribution of artificially engineered features. This graph demonstrates simple frequency counting of three different features.

While individually comparing each feature of fake news spreaders and that of non fake news spreaders, we got some surprising findings.

Figure 2a shows that the distribution of stopword count among non fake news spreaders was much more even than that among fake news spreaders, as around 60 out of 150 fake news spreaders used about 200 stopwords in their tweets. The reason for this may be that the source of fake news is usually much less diverse than that of real news, so that the writing style of fake contents tends to be more uniform, resulting in a similar number of stopwords used by fake news spreaders.

However, a large proportion of fake news spreaders reached a lexical diversity more than 0.4, while much less non fake news spreaders reached lexical diversity scores at the same level (Figure 2b). And the overall lexical diversity of fake news spreaders' tweets was higher than that

of non fake news spreaders, which is contrary to our hypothesis that the writing style of fake contents tends to be more uniform than real contents.

On the other side, as can be seen in figure 2c, there were about 120 fake news spreaders who barely mention other users in their tweets, which was obviously more than the number of non fake news spreaders with similar mention count (around 90). Also, non fake news spreaders had overall more mentions than fake news spreaders, which was surprising as we assumed that, compared with non fake news spreaders, fake news spreaders may tend to mention more users to get attention from others.

We also compared other features of fake news spreaders and those of non fake news spreaders. More detailed figures can be found in the appendix (section 7.2).

4.2 Principal component analysis

To visualize the 2D relationship of the dataset, we performed principal components analysis (PCA) on all the features (see Figure 3). Note that even the dots on the graph were largely overlapped, it does not mean the machine learning classification would fail. Because some machine learning algorithms perform the classification task better in higher dimensions. PCA only demonstrates the relationship of data in 2D perspective. We presented the PCA results of artificially engineered features and the results which included the LSTM features. Since the LSTM features were extracted from trained weights on different neurons, we named them neuron0 to neuron19.

First, we performed the PCA on the artificially engineered features. It is clearly seen the two classes were largely overlapped. Second, we included the extracted weights from the LSTM-based network starting from neuron0. According to Figure 3, there is a clear trend that the two classes of data became more linearly separable as more LSTM features were included. The results of PCA did not change after the first to seventh neuron (which were neuron 0 to neuron6) were included, we only presented the results before the 8th neuron was included.

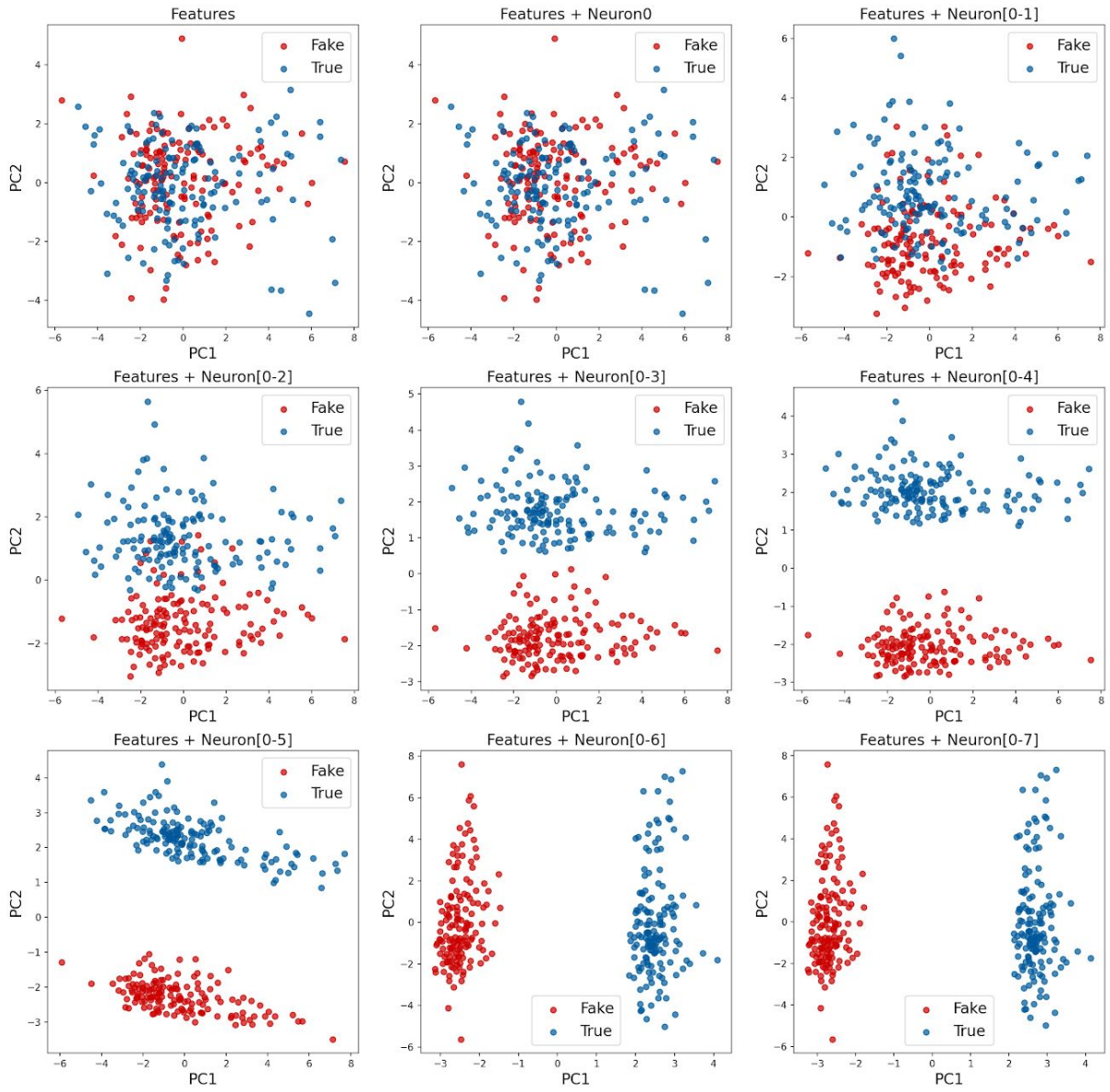

Figure 3. Principal component analysis (PCA). This graph demonstrates the PCA results on all features, including the artificially engineered features and the LSTM features. There are 20 trained weights extracted by LSTM, which were named “neuron” here.

4.3 Comparison of models

Table 1. Comparison of model accuracy

	Accuracy	
Artificially Engineered Features	**English**	**Spanish**
AdaBoost	0.59	N/A
Random forest	0.63	N/A
Logistic regression	0.63	N/A
SVM - linear	0.66	N/A
SVM - poly	0.59	N/A
SVM - rbf	0.62	N/A
SVM - sigmoid	0.65	N/A
Artificially Engineered + LSTM Features		
AdaBoost	1.0	N/A
Random forest	1.0	N/A
Logistic regression	1.0	N/A
SVM - linear	1.0	N/A
SVM - poly	1.0	N/A
SVM - rbf	1.0	N/A
SVM - sigmoid	1.0	N/A
LSTM Features		
AdaBoost	1.0	1.0
Random forest	1.0	1.0
Logistic regression	1.0	1.0
SVM - linear	1.0	1.0
SVM - poly	1.0	1.0
SVM - rbf	1.0	1.0
SVM - sigmoid	1.0	1.0

Here we present an extremely curious case that the models were able to achieve 1.0 accuracy on the validation dataset after adding the extracted features from LSTM. Table 1 presents the comparison of models accuracy. First, it is important for us to specify that all of the exploratory data analysis and LSTM-based network were performed on English dataset only. We do not have any artificially engineered feature of Spanish dataset. Therefore, two sections in Table 1 were marked as N/A.

In the current project, we compared a binary classification of fake news identification on several models, including: AdaBoost, Random Forest, Logistic Regression, and different kernels of SVM. With our artificially engineered features, the accuracy was able to achieve 0.66 at the SVM linear. However, after appending the features extracted from the LSTM-based network, the model was able to achieve 1.0 accuracy on the validation set. We also tested the model on LSTM features only, turns out the accuracies on all models were 1.0 as well. Out of curiosity, we decided to test the **same** LSTM features on Spanish dataset. And the accuracies were also 1.0. As stated before, the LSTM-based network did not train on any Spanish data. At first, we thought the results were due to programming errors. However, after carefully excluding possibilities, the puzzling results remained (see *Discussion*).

4.4 Feature importance

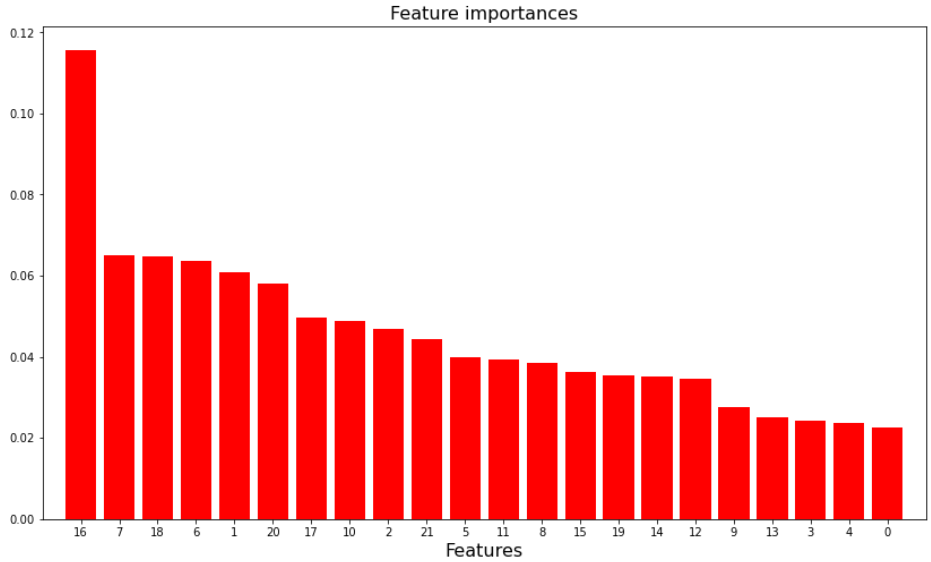

Figure 4 Feature Importance (artificially engineered features). This graph demonstrates the ordered feature importance in the Random Forest model. The ticks in X axis represent the order of training dataframe.

Figure 4 shows the importance of each artificially generated feature after fitting the Random Forest model. The most important three features are the mean of sentiment values, lexical diversity and the mean of subjectivity values. We calculated the average sentiment score among the all fake news spreaders, which was 0.07, lower than that of non fake news spreaders (0.11). This may suggest that fake news spreaders tend to use more negative expressions, which corresponds to our hypothesis. The average lexical diversity of non fake news spreaders was 0.38, while that of fake news spreaders was 0.44, which was out of expectations as we assumed the opposite. Similarly, we found it surprising as the average subjectivity score of non fake news spreaders was slightly higher than that of fake news spreaders (0.31 vs. 0.29).

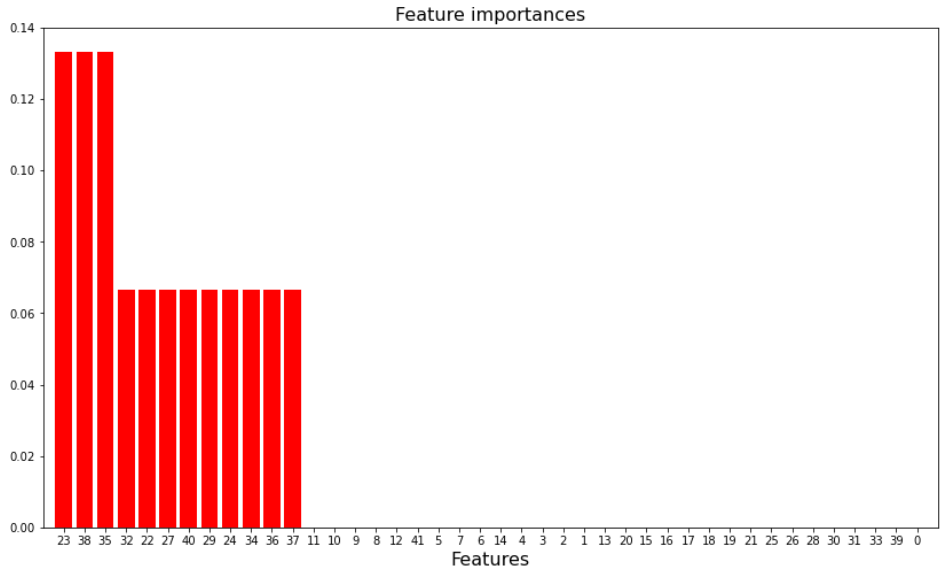

Figure 5. Importance of all features on Random Forest model

Figure 5 shows the importance of all the features after the fitting random forest model, i.e. the 22 artificially generated features and the features extracted in LSTM model. The most important 3 features are those captured by neuron 1, neuron 16 and neuron 13.

5. Discussion

In the current project, we analyzed the tweets posted by both true and fake news spreaders. We generated 22 features based on our linguistic knowledge and hypotheses toward the fake news spreaders. Meanwhile, a LSTM-based neural network was trained directly on the entire corpus. Lastly, the artificially engineered features and the weights extracted from the LSTM-based network were combined as input for other classification algorithms. Extremely curious modelling results, which might indicate the outstanding effectiveness of LSTM features, were presented.

5.1 Unreasonable performance of extracted LSTM features

The features generated by the LSTM-based network were unreasonably effective for the Twitter fake news spreader classification. At first, we believed it was due to the programming error. Since the data structures were different for LSTM and other models (see *Methods*), an obvious possibility is that there was something wrong regarding the separation of the training dataset, causing the validation dataset were accidentally mixed into the training stage of the LSTM-based network. We carefully excluded this situation by a) making sure the random seed was applied in all train-validation splitting functions. We ran the classification models in a local machine while utilizing Google Colab GPU for the LSTM training. Being afraid that the same random seed could perform differently in two environments, the index was compared and made sure to be the same on two machines after splitting. B) Cutting every chance to access validation data in the Google Colab environment. The data was split ahead into train and validation sets. Only the training data was sent in the google drive which Colab could fetch. C) When training classification algorithms, making sure the only difference in the codebase was if the LSTM features were included. Also, visualize the values of the features by directly copy and paste the values instead of saving them in a variable. Despite being careful not to mix the training and validation data, the puzzling performance of LSTM features remains. In sum, we continue to hold a skeptical attitude towards the codebase. There could be some other errors in the codebase. However, the detection of the errors could be beyond our knowledge.

Suppose there was no error occurred in our codebase, which means that including LSTM features could actually boost the performance of other classification algorithms. That might be

due to the fact that the LSTM-based network took the entire corpus into evaluation. And by doing this, the network successfully captured some characteristics of fake news spreaders which were abandoned when we generated the artificially engineered features. Meanwhile, such characteristics are language-independent because the features yielded the same results on Spanish dataset as well.

5.2 Difficulties of fake news spreader identification

First of all, one of the main difficulties is that of labeling and the question of what "fake" actually means and what is fake or true and when it is not. To illustrate this, Ruchansky et al (2017) found out that when an adult sees a fake news article, it will be assessed as true in 75% of the cases, for high school students the percentage is even at 80%. In the introduction we've stated different studies that showed lower numbers. Still, those were far over 50% (62%), showing that humans are not able to reliably detect fake news themselves, hence the need for computational detection methods. But as we can see in this paper, such computational methods are still far from being perfect. Furthermore, alongside the developing methods for detecting fake news, also new methods for spreading and writing fake news arise, making it difficult to stay ahead with security measurements to prevent the spreading of those news.

Apart from that, a second difficulty is to determine authorships. There might be accounts that were created to only serve the need of creating and spreading fake news, which are then easier to eliminate. However, there might also be real users, so actual existing persons, that share fake news a lot which might be because they think it is correct information and are not able to distinguish fake from authentic news (see above numbers regarding human's ability to identify fake news). The paper shows that even computational methods do perform better than the humans studied in the papers consulted for this work: linear SVM gave an accuracy of 0.66 and the Random Forest classifier performs lower and is at 0.63 accuracy. These numbers show the challenges that fake news are for computational and automatic detection.

6. Bibliography

Literature

1. Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, *52*(1), 1-4.
2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805.
3. Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). V*isualizing higher-layer features of a deep network*. University of Montreal, 1341(3), 1.
4. Granik, M., & Mesyura, V. (2017, May). Fake news detection using naive Bayes classifier. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (pp. 900-903). IEEE.
5. Long, Y., Lu, Q., Xiang, R., Li, M., & Huang, C. R. (2017, November). Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 252-256).
6. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781.
7. Rangel, F., Rosso, P., Ghanem, B., & Giachanou, A. (2020). *Profiling Fake News Spreaders on Twitter* [Data set]. Zenodo. Available on http://doi.org/10.5281/zenodo.3692319
8. Ruchansky, N., Seo, S., & Liu, Y. (2017, November). Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (pp. 797-806).
9. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, *19*(1), 22-36.
10. Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & de Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*.

11. Tschiatschek, S., Singla, A., Gomez Rodriguez, M., Merchant, A., & Krause, A. (2018, April). Fake news detection in social networks via crowd signals. In *Companion Proceedings of the The Web Conference 2018* (pp. 517-524).
12. McCarthy, P.M., & Jarvis, S. (2010). MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment, Behavior Research Methods, 42(2): 381-392

Coding literature and material

13. Kyle, K. (2020). lexical-diversity 0.1.1. MIT license. https://pypi.org/project/lexical-diversity/
14. Bird, S. (2020). Natural Language Toolkit 3.5. Apache Software License. https://www.nltk.org/
15. Explosion AI. (2020). spaCy 2.2.4. MIT license. https://spacy.io/
16. Loria, S. (2020). Textblob 0.15.3. MIT license. https://pypi.org/project/textblob/
17. Li, S. (2019). Multi-Class Text Classification with LSTM. Medium. https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17

7. Appendix

7.1 Exam questions

Imagine you want to spread some fake news. Imagine that all social media have installed trust indicators and once a user publishes some fake news, the user is kicked out. How would you avoid being kicked out when spreading your own fake news?

From what we analyzed in this paper, we found the most important feature for fake news spreader identification was mean of sentiment values, and fake news spreaders tend to use more negative expressions. Thus, we can avoid too much negative content while creating fake news. Also, we consider the following account information which is not available in this present dataset will be important for fake news spreader identification.

- Time: fake news spreaders post mostly in working hours
- Interaction: fake news spreaders could follow people but seldom interact with friends (such as commenting under the posts)
- Account name: fake news spreaders have simple (maybe similar patterns) account name
- Trend: fake news spreaders post similar trends in a very similar time
- Topics: fake news spreaders post a lot of political news and click bait-like topics

Therefore, we can register an account with a real name, and form a social network by adding other users and interacting with them. While editing our own fake news, we need to avoid obvious clickbait titles and too much political topics, or we can substitute frequently used words in political news by less common ones. And finally, when we post our fake news, we need to avoid doing it in working hours and posting similar trends at the same time with others.

We suppose that in some cases, you may have not explored some possibilities because of limitations in terms of computer power and data size. Suppose that you have unlimited power and data. What do you think could be useful in order to improve your current model?

At first, we would like to try the bag-of-words (BoW) and the word embedding methods in order to vectorize the corpus directly. However, both methods used computational powers beyond our reach. Since there are 30,000 tweets in the corpus, BoW methods could make the classification algorithms we used to become computationally intensive. The BERT model transforms a word to a representation in 768 dimension. According to a rough statistic, it took around 5 minutes to just transform a single sentence containing 6 words. The lack of computational power stopped us from conducting these two approaches.

However, we think to transform the corpus data into a very high dimension and sparse representations could be a good idea (not limited to BoW or BERT). Consider human brains as a benchmark. If a well-trained human expert could perfectly detect the fake news spreader, the human must have developed sophisticated mind representations to capture all the characteristics

of fake news spreaders. Taking the scale of brain neurons into consideration. These mind representations, which might be operated by neurons, must have extremely high dimensions. Also, one of the reasons that humans can identify fake news might be thanks to the knowledge or "common sense" that was stored in the brain formerly. By that knowledge, we are easily found skeptical if the information is counterintuitive. If we have unlimited computational power, which enables a large model to be trained on infinite data, there's a high chance that the model could develop a "common sense" model. Normally, if one knows the correct keywords, fake news (or disinformation) could be resolved by searching just 1 or 2 topics. This kind of searching could be executed automatically if we have a common sense model.

7.2 Distributions of artificially engineered features

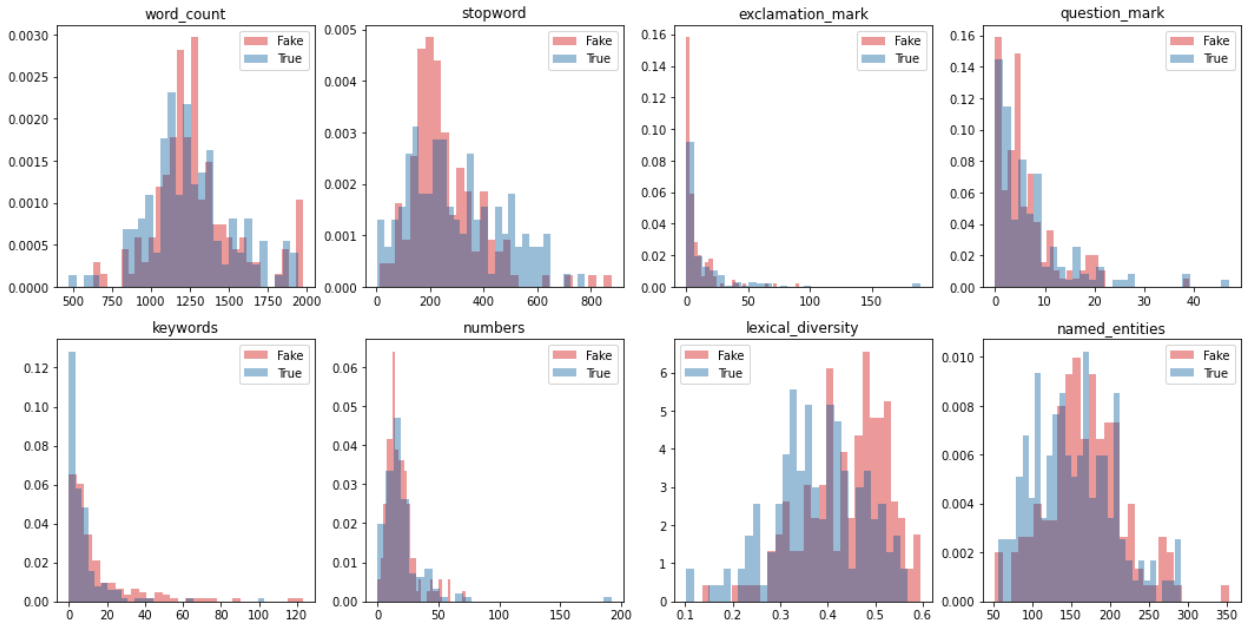

Figure 6. Frequency distribution of lexical features.

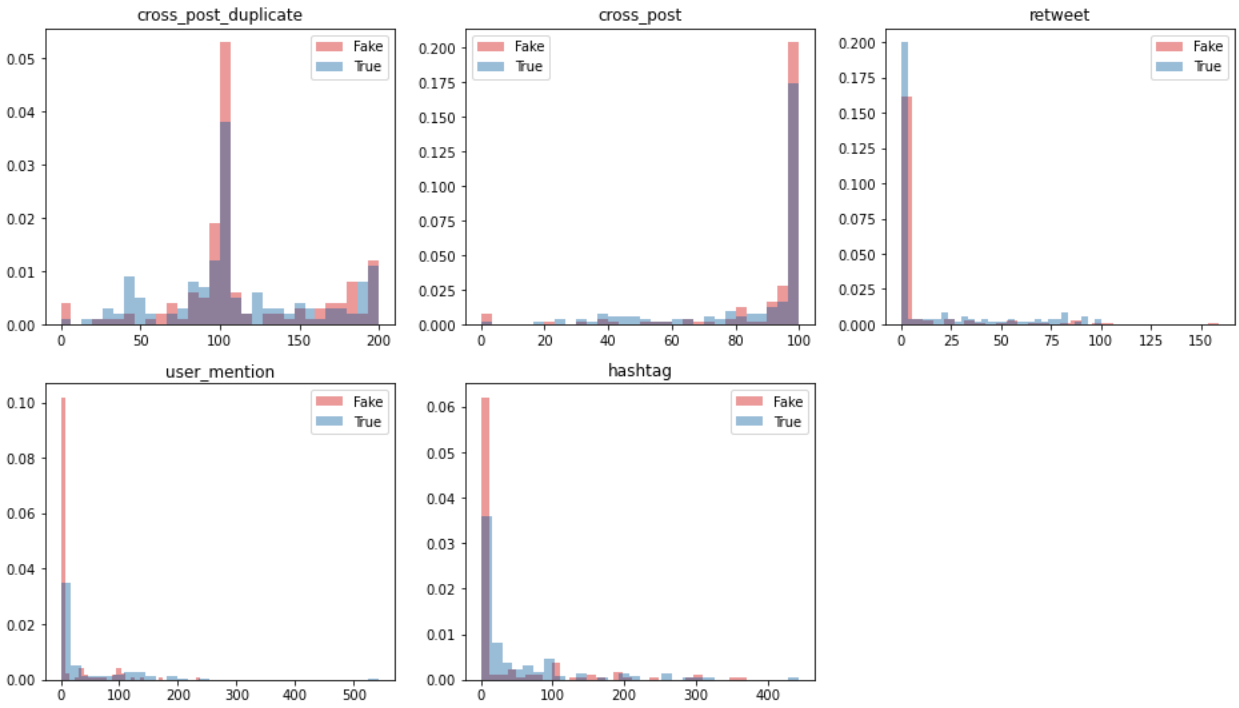

Figure 7. Frequency distribution of Twitter user behaviors features.

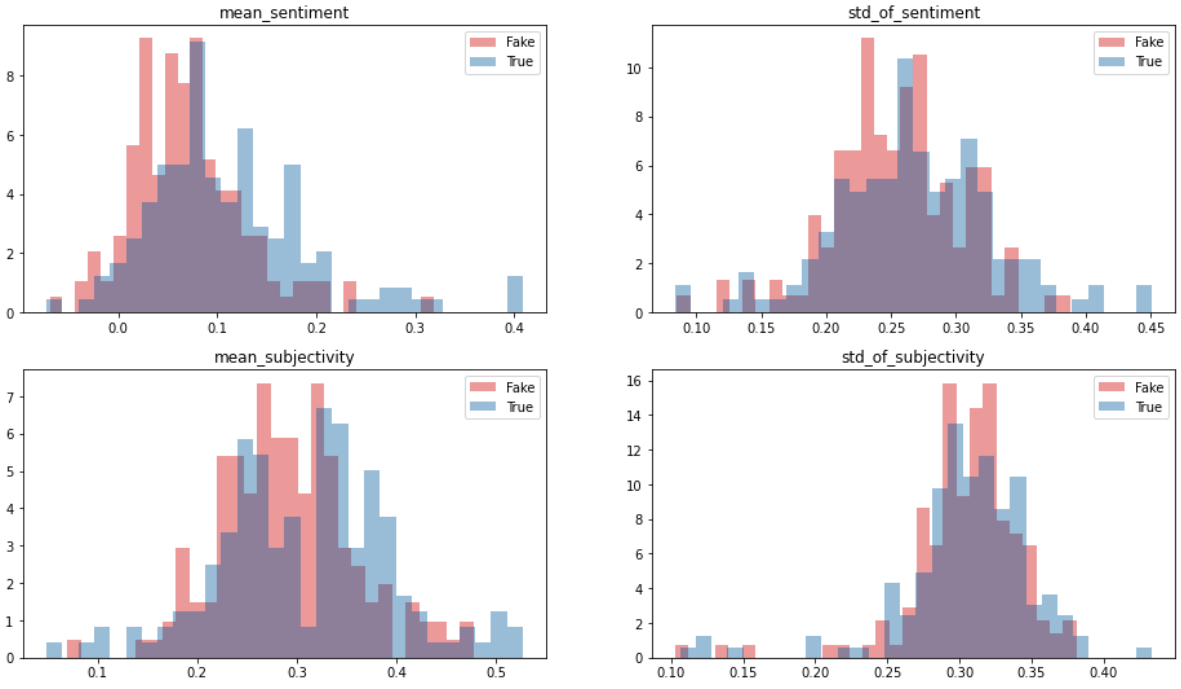

Figure 8. Frequency distribution of sentiment features.

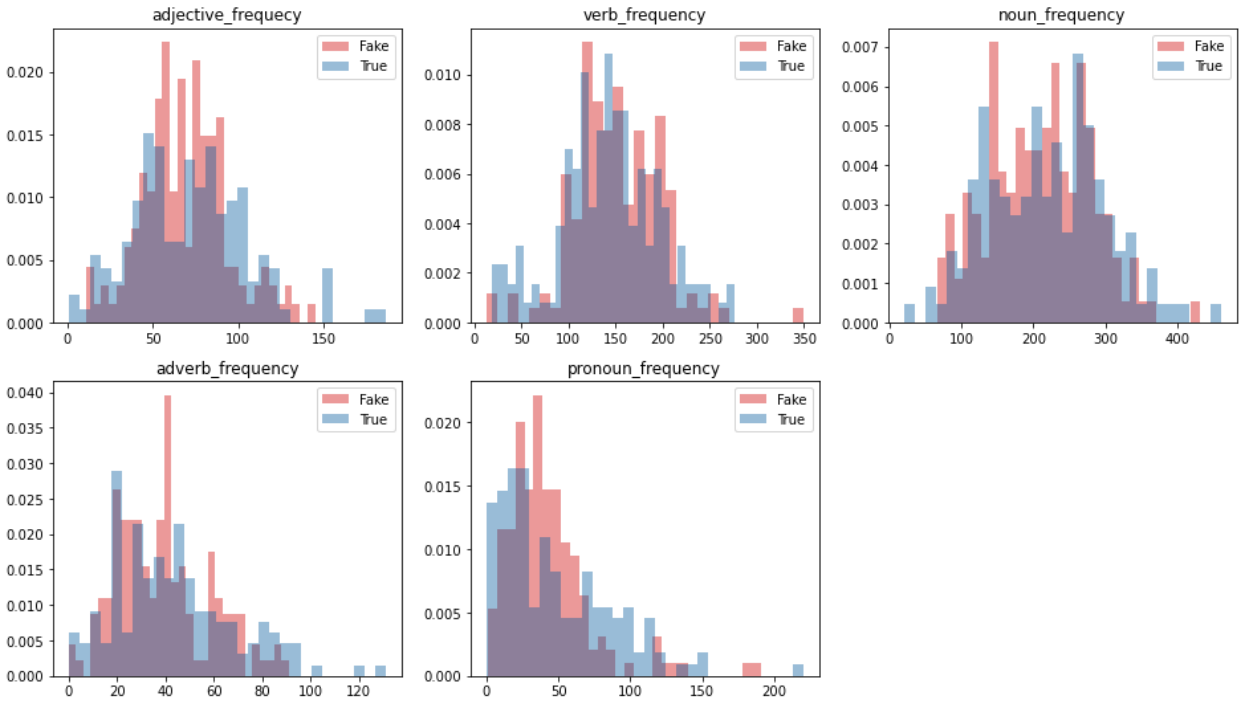

Figure 8. Frequency distribution of syntactic features.

7.3 Table of contribution of each group member to this work

	Mengran Chen	Lena Burkel	Ting-Yu Kuo
Coding			
Artificial feature generating	x		x
LSTM neural network modelling			x
Classification using Machine learning models			x
Report			
abstract	x		
introduction		x	
dataset description	x		
methods	x	x	x
results	x	x	x
discussion		x	
exam questions	x	x	x