Automated Fact Checking (COMP0084 Course Project)

Student Number: 14019634
University College London

1 INTRODUCTION

Due to a large amount of online misinformation spreading through the years, automated fact checking has become a popular research area. This project develops and uses different methods of information retrieval and data mining techniques to assess the accuracy of a claim. Three main steps are evaluated in this project consisted of: relevant document retrieval, evidence selection and claim veracity prediction as well a thorough literature review. The publicly available Fact Extraction and Verification (FEVER) dataset will be used to solve the research oriented questions. This project consists of solving several subtasks starting from simple text statistics, vector space document retrieval, probabilistic document retrieval, sentence relevance and evaluation and truthfulness of claims. To identify sentence relevance a logistic regression model was created and for the truthfulness of claims multiple neural nets were created to test the quality and performance of the architecture.

2 TEXT STATISTICS

The proposed first task is to count frequencies of every term in the collection of documents. Relatively a simple task, by iterating over all the documents each "*text*" field was cleaned by making all the text to lowercase (so no duplication occurs), removing all punctuation as well as splitting the terms. The frequency of every word was stored in a dictionary and the subset of some of the words can be viewed from the curves of term frequency in Figure 1.

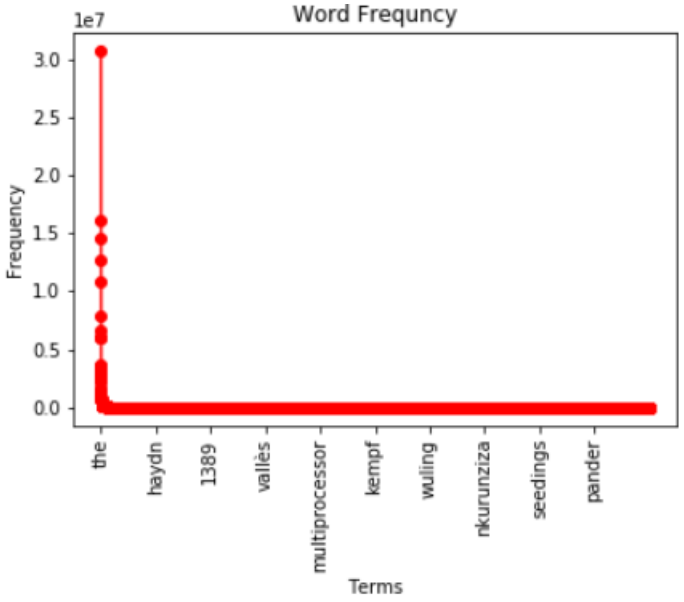

Figure 1: Word frequency of around 15k terms and their frequencies, some terms of the corpus are shown on the x-axis

Additionally, Figure 2 verifies Zipf's law as the frequency of any terms is inversely proportion to it's rank, given that a large sample of terms is used. According to Zipf's law, by multiplying the ranking position of a term with its probability, it should result approximately to a constant. For the English Language this is around 0.1. Five values randomly uniformly chosen from the first 400 words of the corpus that can be viewed in table 1. As observed most values are close to the constant 0.1 and the average of the values is also approximately around 0.1.

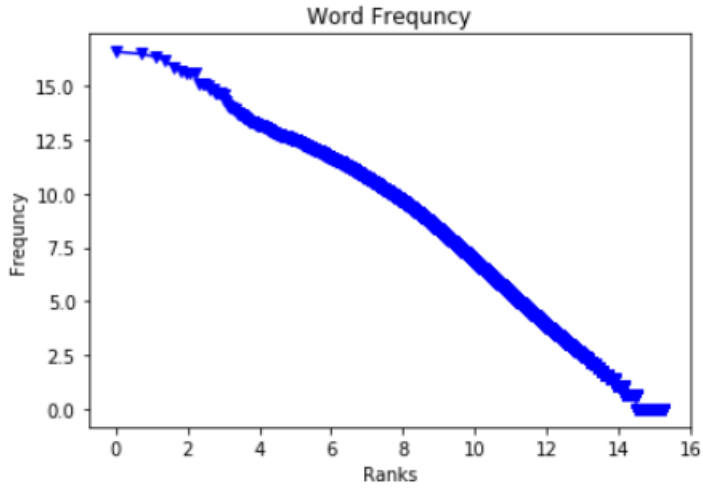

Figure 2: Zipf's law demonstrated, the frequency of a word being inversely proportional to its rank, both axis being in logarithms

Table 1: Values of the parameters for Zipf's law

Term	Values
298	0.0992
287	0.0984
317	0.1004
228	0.0.0935
8	0.1053

3 VECTOR SPACE DOCUMENT RETRIEVAL

This section focuses on extracting the TF-IDF representation of the 10 claims [75397, 150448, 214861, 156709, 129629, 33078, 6744, 226034, 40190, 76253] and all the documents respectively based on the document collections. After creating this representation, you can compute the cosine similarity between the document and the claims.

Before diving into to calculating the TF-IDF representations, for computational efficiency, the documents are represented based on the words that have effect on the cosine similarity calculations. To do this a function stored all the unique words from the claims and removed the *filler* words that were identified as not important. The filler words were often **stop words** which were the following: *with, the, is, a, of, includes, and, an, based, on, their, at, has, new, in, there.*

To make querying faster, the data structure **Inverted Index** was implemented. This is a general name for class of structures and it is *inverted* because documents are associated with words, rather than words with documents. This is to support full text search over a set of documents. Essentially, it is a large table where there is an entry for a unique word in all the documents processed, along with list of key pairs of document id and the frequency of the term in the document. Each text for the Inverted Index were split as well as cleaned by making all letters lowercase and removing all punctuation.

After the creation of the Inverted Index to understand how important a word is to a document in a corpus the term frequency-inverse document frequency (TFIDF) was developed. To develop this the TFIDF is separated by two terms. The first term computes the normalized term frequency i.e how frequent a term occurs in a document. The second term inverse document frequency computed as the logarithm of the number of documents in the corpus divided by the number of documents where the specific term appears, i.e how important a term is. Then the final TFIDF is created by the product of these quantities.

The cosine similarity is a measure of similarity between two vectors that measures the cosine of the angle between them. Such vectors can represent a document and the greater the value of θ, the lower the value of $cos\theta$, hence the less similarity between the two documents (in our case the similarity between the documents and the claims) and vice versa. For a given claim, the cosine similarity for each document for the given claim can be viewed in the CSV file. A sample of the results can be seen below in table 2:

Table 2: Cosine Similarity of Documents for Claims

Claim	Document	Cosine Value
75397	Nikolaj_Coster-Waldau	0.8564
150448	Giancarlo_Volpe	0.7172
214861	Baroque	0.7090
156709	Alfonso_Gagliano	0.7120
129629	Gideon_Raff	0.6715

4 PROBABILISTIC DOCUMENT RETRIEVAL

4.1 Query-Likelihood Unigram Model

This section will explore the query-likelihood unigram language model based on the document collection to establish the five most similar documents in each of the 10 claims. In addition to that three types of smoothing is applied to the query-likelihood language model to improve the results.

Firstly, to construct query-likelihood unigram language model, a language model is created for every document in the collection (i.e count the frequency of the word). After the creation of the language model, it is possible to rank documents by the probability of "*generating*" the claim. This is to compute the probability of generating one word in the claim and then multiplying the probabilities of all words in the claim. The formula used to demonstrate this is as follows:

$$P(Q|M_D) = P(q_1...q_k|M_D) = \prod_{i=1}^{k} P(q_i|M_D)$$

where M_D is the language model for every document D and Q the query or in our case the claims. It is important to note that if one term in the claim does not exist in a document the final probability will result in returning zero. Due to this some claims did not have any returned relevant documents. Therefore, in the final CSV final the claims that did not return any relevant documents are randomly filled by random documents and highlighted. A sample of relevant documents for some of the claims can be viewed below and for the full set please refer to the applicable CSV file.

Table 3: Query-Likelihood Unigram Model Similarity Scores

Claim	Document	Score
214861	History_of_art	-173.8191
129629	Homeland_-LRB-TV_series-RRB-	-97.1496
33078	Boston_Celtics	-78.0589
6744	Katherine_Orrison	-29.5803
40190	Cyndi_Lauper	-113.7738

To get around the problem of query words missing from documents and returning a probability of a zero, smoothing techniques can be applied. Since document texts are a sample from the language model, missing words should not have zero probability of occurring. Smoothing is a technique for estimating probabilities for missing words by discounting the probabilities estimates for words that are seen in the document text. By doing so every term in the document will have some probability of occurring in every document. Below three different smoothing techniques are implemented and the five most similar documents for 10 claims can be viewed in the associated CSV file.

4.2 Laplace Smoothing

Starting off from Laplace Smoothing, the simplest form, the algorithm involves counting events in the observed data, add 1 to every count, re-normalize to obtain probabilities. The formula below is implemented:
Max Likelihood Estimates:

$$\frac{m_1}{|D|}, \frac{m_2}{|D|}, ..., \frac{m_{|V|}}{|D|}$$

where D is the document, m is the number of occurrences of words with $\sum_i m_i = N$ and where $|V|$ is the number of unique words in

the vocabulary size. Then the Laplace estimate could be calculated as:

$$\frac{m_1 + 1}{|D| + |V|}, \frac{m_2 + 1}{|D| + |V|}, ..., \frac{m_{|V|} + 1}{|D| + |V|}$$

A sample result of the Laplace smoothing can be seen in table 3, the full result can be viewed in the CSV file:

Table 4: Laplace Smoothing Scores

Claim	Document	Score
75397	Jim_Paymar	-30.5318
150448	Giancarlo_Volpe	-97.1496
214861	History_of_art	1.3598
156709	All_You've_Got	-15.2659
33078	Boston_Celtics	6.3303

4.3 Jelinek-Mercer Smoothing

A key problem with Laplace smoothing is that it gives too much weight to unseen terms. Discounting treats unseen words equally which is problematic as some words are more frequent than others. Interpolating methods use the idea of using *background probabilities* which reflects the expected frequency of events. The background probability can be estimated for unseen words as:

$$P(w|C)$$

where $P(w|C)$ is the probability for word w in the collection language model for collection. To estimate for words that occur we can estimate:

$$\lambda P(w|D) + (1 - \lambda)DP(w|C)$$

where λ is a parameter.
Jelinek-Mercer Smoothing uses the above idea by setting λ to be a constant, independent of document, query. For the purpose of this project $\lambda = 0.5$ gave the best performance by trial and error. A sample of the results can be viewed in table 5 and the full documents can be seen in the CSV file:

Table 5: Jelinek-Mercer Smoothing Scores

Claim	Document	Score
75397	Shakespeare_Company	-7.2993
150448	Roman_food	-8.5941
214861	Topless_-LRB-film-RRB-	-6.6998
156709	Certified_Accountant	-12.1040
33078	List_of_role-playing_video_games	-7.7298

4.4 Dirichlet Smoothing

The problem with Jelinek-Mercer Smoothing is that if you proceed the smoothing on longer documents, they will provide better estimates. Dirichlet uses a slightly different approach that makes smoothing depend on the sample size.

$$\lambda = \frac{N}{N + \mu} \quad + \quad (1 - \lambda) = \frac{\mu}{N + \mu}$$

where N is the document length, i.e the length of sample and μ is a constant, where it is the number of word occurrences in the document divided by the number of total documents. Some sample example of the Dirichlet smoothing can be seen in table 6 and the full results will be available in the according CSV file:

Table 6: Dirichlet Smoothing Scores

Claim	Document	Score
75397	Steam_Navigation_Company	-2.1951
150448	History_of_Rome_-LRB-disambiguation-RRB-	-2.4863
214861	Music_of_Asia	-1.6654
156709	Certified_Accountant	-2.5658
33078	Yampa_River_Botanic_Park	-2.1422

5 SENTENCE RELEVANCE

For a claim in the training subset and the retrieved five documents for the claims which was based on cosine similarity, the claims and sentences were represented in these document based on the Wor2Vec embedding method. Before converting the claims and sentences into embeddings, Gensim was used to get an accurate pre-trained word embedding using the *glove-wiki-gigaword-100* data-set.

The claims were retrieved as previously mentioned and the relevant sentences were split by *lines* and stripped similarly to the claims. To create the word embeddings the claims and the sentences were cleaned and created and the size of the features were chosen to be 100 for both claims and sentences. This enabled a good amount of feature selection and as well as providing a suitable computation time. The claim and and sentences were then concatenated creating a size 200 features. Additionally, the words that are not present in the pre-trained model were simply ignored. This in theory would not effect the sentence relevance too much as removing one word is very unlikely to have a huge effect on the sentence relevance.

To feed the model the relevant sentences, in the training data the "evidence" field records all the relevant sentences, the code implemented takes all the relevant sentences for each claim. At the same time negative sampling is also used to collect the same number of irrelevant sentences as relevant ones from the same claim. The number of irrelevant sentences and relevant sentences are the same to make the class balanced for both training and development sets (50:50 split). For computation efficiency 500 relevant and irrelevant sentences were stored, using a total of 1000 sentences.

A logistic regression model was implemented trained on the subset to identify five relevant documents to the claims in the development data and select the five most relevant sentences for a given claim within these documents. Initially the weights were all initialised to ones and for every iteration the model randomly chose from the uniform distribution to sample the weights. After running the model for around 100 iterations the best weights were returned.

Using the first 10 verifiable claims in the development set, the training accuracy was around approximately 83.6%. Since the classes are balanced one evaluation metric used that fairly fit this task was accuracy. The performance of the model was approximately around 56%. The accuracy is low because the number of examples used per claim were very low (5 examples). By increasing the number of claims this does not really help in improving the model, whereas if more examples were trained per claim the accuracy could be improved.

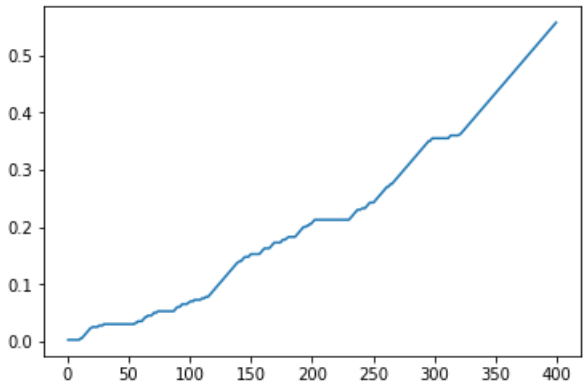

Figure 3: Accuracy over 400 iterations, where x axis shows the iterations and y axis shows the accuracy

The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error. It is always a challenging process as a very small learning rate could result very long training times which may eventually get stuck and too large of a value may result in learning sub-optimal weights values. For this project a variety of learning rates were explore ranging from big values such as 1 to really small values such as 0.0001. The balance came with the value of 0.01 as it gave the best configuration for the network.

6 RELEVANCE EVALUATION

To implement methods to compute Recall, Precision, F1 metrics four calculations need to be done: True Negative, False Negative, False Positive, True Positive. The values for these calculations from the logistic regression were:

$$FalsePositive : 59,\ TruePositive : 82$$

$$TrueNegative : 141,\ FalseNegative : 118$$

6.1 Precision

The precision score of the model was 0.58.
Precision is calculated by:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

The denominator indicates the total predicted positives. Hence, it show how precise/accurate our model is out of the predicted positives. Precision is normally good in a situation where the costs of False Positive is high.

6.2 Recall

The recall score of the model was 0.41.
Recall is calculated by:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Recall calculates how many of the actual positive our model captures i.e through the labeling of True Positives. Recall should be considered a good metric when there is a high cost associated with False

6.3 F1

The F1 score of the model was 0.48
F1 is calculated by:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1 score represents a balance between the recall and precision score and there is an uneven class distribution.

7 TRUTHFULNESS OF CLAIMS

To begin uncovering truthfulness of claims, the claims were first filtered out the “NOT ENOUGH INFO" claims and only the “SUPPORTS" and “REFUTES" claims in the train and dev datasets remained, making the classification binary. Similarly to question 4, for the “SUPPORTS" claims, the labels marked as 1 and for the “REFUTED" labels, marked as 0. The data fed to the neural net was again balanced with the equal number (50:50 split) of claims of “SUPPORTS" and “REFUTED" where it was fed into the neural net.

7.1 Motivation

Keras Sequential models were opted in creating neural nets as they provide a good basis for creating neural nets. The sequential model is a linear stack of layers. The sequential models provide a flexible way of creating different neural architectures as well as an easy way of experimenting with different hyperparameters such as optimizers, learning rate as well as providing a variety of metrics. The main metrics used for the purpose of this project was *Accuracy*.

7.2 Performance

A simple single layered sequential neural net was created using 2D layers, such as “*Dense*". The “Dense" layers allow all nodes in the previous layer to connect to the nodes in the current layer. Additionally, the relu activation function as it is the most commonly used in neural nets as well as sgd for the optimizer. The accuracy achieved by the model after 150 epochs was **51.80**%. This is not an ideal prediction given the task is binary. This is predictable as the training data holds a small sample/examples from each claim. Deep Neural nets require large amount of training data to perform well as well as longer epochs, as small datasets can be either misleading or non-representative with respect to the underlying trends. Having said that part of the low performance of the model is due to

architecture being only single layered. In section 8 different sequential models will be explored that can improve the single layered approach.

8 LITERATURE REVIEW

With the advancement of technology, rise of social media and the ease of communication between individuals and communities, misleading information in everyday access have made it challenging to identify trustworthy sources. In this section a literature review regarding fact checking and misinformation detection will be conducted identifying a critical analysis of existing models and methodologies and the drawbacks of such methods.

8.1 Automated Fact Checking

The 21st has brought an age of data and information overload. As a result an increase of spread of misinformation has also been visible. Perez-Rosas, et al., (2017) focuses on automatic identification of fake content in online news [6]. To do so they propose *twofold* which first introduces two novel datasets for the task of fake news detection. Previous datasets are typically focused on only one domain and to address such a shortcoming the datasets are created covering several domains that model the properties of fake news. Analysis is then conducted on the identification of linguistic differences in fake and real news, with descriptions of collection, annotation as well as the validation process. Secondly, by applying machine learning fake news detectors using the linguistic features achieved accuracies of up to 76% which were compared to the performance of an empirical human baseline. The proposed linguistic driven approach highlights that it would be relevant to view lexicon, syntactic and semantic level of a news item. Although, linguistic features provide a promising framework in automatic detection of fake news, other *meta* features should not be neglected such as comments on an article and features from different modalities (i.e use of machine vision for the visual makeup of a webpage) as well as other computational methodologies that are already established to fact verification.

Although automatic fact checking shows promising results, both academic researchers as well as professional established fact checking organisations (Graves 2018) agreed that fully automated fact checking remains a distant goal [2]. The most effective automatic fact checking tools are the ones that help fact-checkers respond more promptly and effectively to disputes, online rumours and other different types of misinformation. Additionally, new tools are being developed to match claims against previous work, by automatically highlighting related fact-checks and any other relevant information to help fact-checkers intervene quicker.

However, automated responses without any human intervention that could be applicable at scale has been show to be limited [2]. Although, progress is being made, techniques that try to imitate humans in understanding claims and checking them against their respected references are constrained due to the lack of data as well as the current state of technology. One way of approaching to solve this task could be to understand various unstructured approaches, although, it is unclear how effective they can be.

8.2 Modelling Frameworks

The Internet and online social media have provided a proliferating ground for the spread of misinformation and hoaxes. Modelling frameworks have been developed through multiple research to tackle this problem. Tambuscio et al., (2015) studies the idea of diffusion of hoaxes [4]. Using traditional mathematical modeling hoaxes are treated as viruses where a user can become infected to one if exposed and also a spreader. Upon verification, users can also turn into *non-believers* and spread the information as *believers*. Their model consists of focusing on four variables: spreading rate, gullibility, probability of verifying a hoax and the likelihood of forgetting your own current belief. By simulating the models on real networks, many of the variables highlight the threshold for the fact checking probability that can confirm removal of a hoax from a network, with the most outstanding feature being gullibility and the forgetting likelihood probability. The drawback of this model is that it all the agents have the same gullibility and verification probability whereas in the real world, individuals will have different aptness to believe claims that are along their own world view beliefs.

Computational models have been extremely helpful in assessing the veracity of claims, however important, it is also a very timely problem. In practise automatic fact checking tools and models have to be explainable, accurate and fast. Nguyen et al., (2018) states that although prediction accuracy is an important factor of, transparency of the models should also be clear for users to trust and integrate their own knowledge with the system [1]. Hence, the proposed idea is a novel probabilistic graphical model which merges machine learning with crowd annotations. The model improves performance and interpretability for predicting claims as well as providing fast parameter estimations. The results based on a small user study showed strong transparency, strong predictive performance as well as improving user satisfaction and trust.

8.3 Leveraging the Crowd

Not all frameworks need to be solely mathematical or only technology dependent. Choy and Chong (2018) develop a framework that allows a lay person to identify potential fake news, without the need of any complex statistical means or interpretations [5]. The three dimensional framework simplifies the identification of fake news consists of:

- Strong positive and negative words
- Length of the title
- Consideration and selection of verbs, names, adjective and numbers

Similarly, online social networking platforms are using their users or in other the words the crowd to reduce the spread of fake news by allowing them to *flag* any stories or information they perceive to be a hoax or fake. If a certain piece of information receives sufficient number of flags, the content is sent to trusted third party organisation for fact checking. However, the fact-checking process can be very costly as well as the trade off between flags and exposures, the procedures require very careful examinations and smart algorithms which are not developed yet. Kim et al., (2017) introduces a flexible representation framework of marked temporal point processes as well as an online algorithm to select which content/stories should

be sent for fact checking and when to do so efficiently [3]. The algorithm was experimented on real datasets from Twitter and Weibo and proved to be effectively reducing the spread of misinformation. The framework however assumed that every person within a crowd is equally good or bad at flagging misinformation. This idea could be changed to have less assumptions about an individuals abilities or trustworthiness, which is closer in the real world, to design algorithms that are robust to unpredictable behaviours from the crowd.

In summary the technology that exists now enables us to develop, scale up and speed up fact-checking drastically better than before. Many organisation and research institutes are putting practical automated tools alongside fact-checkers. Many fact checking projects have been undertaken in various places around the world and open standards and design practises for automated fact checking should come to fruition in the near future so that a shared infrastructure is created. Additionally, international collaboration is a must to be able to create systems that work in multiple languages.

9 MODEL IMPROVEMENT

This section will highlight improvements for the machine learning models implemented above. A more complex neural architecture is proposed as well as techniques that can be used to increase accuracy of the models.

9.1 Architectural Improvement

To improve the previous neural net, the first and most common way is to add more layers, so that all the nodes in previous layers can be connected to the current layer which will improve learning. Other improvements made to the model include changing the optimiser from “sgd" to “Adam" as well as increasing the number of epochs the model runs for. Finally, dropout was added in between one of the “dense" layers. Dropout can be configured to ignore a certain percentage of units during training at random, which results to a reduced network. Dropout is useful as it helps to prevent overfitting, since the neuron in a fully connected layer can develop co-depnedencies amongst each other during training. By increasing the number of layers and epochs, changing the optimizer as well as adding dropout, the model managed to achieve a **56%** accuracy, which is an increase of **4.2%**

9.2 Further Improvements

Other complex sequential models such as a LSTM was also implemented to improve performance. LSTM’s are a special kind of RNN are able to to learn long term dependencies. They are able to remember information for a long period of time. The implementation of all the models can be seen in jupyter file provided.

An improvement that could be done when cleaning the words and the documents would be splicing. Splicing is able to get rid of many words that do not add real value to the sentence as well as improving the computation time. Additionally, if the claims contained many more relevant sentences to be trained on, the accuracy of the model would have been further increased.

REFERENCES

[1] Matthew Lease Byron C. Wallace An T. Nguyen, Aditya Kharosekar. 2018. An Interpretable Joint Graphical Model for Fact-Checking from Crowds. (2018).

[2] Lucas Graves. 2018. Understanding the Promise and Limits of Automated Fact-Checking. (feb 2018).

[3] Alice Oh Bernhard SchÃűlkopf Jooyeon Kim, Behzad Tabibian and Manuel Gomez-Rodriguez. 2017. Leveraging the Crowd to Detect and Reduce the Spread of Fake News and Misinformation. (nov 2017).

[4] Alessandro Flammini Filippo Mencze Marcella Tambuscio, Giancarlo Ruffo. 2015. Fact-checking Effect on Viral Hoaxes: A Model of Misinformation Spread in Social Networks. (may 2015).

[5] Mark Chong Murphy Choy. 2018. Seeing Through Misinformation: A Framework for Identifying Fake Online News. (mar 2018).

[6] Alexandra Lefevre Rada Mihalcea VerÃşnica PÃlrez-Rosas, Bennett Kleinberg. 2017. Automatic Detection of Fake News. (aug 2017).