ALEXANDRU IOAN CUZA
UNIVERSITY of IAŞI

Technical Report

Fake News

Bardita Larisa-Ioana
Chirica Demetra-Bianca
Chisa Daniel
Miron Robert-Andrei
Susan Stefan-Claudiu

Cuprins

1 Problem presentation

For as long as we've known, fake news have been actively spread around us, be it called rumour, diversion, misinformation or propaganda.

The main problem remains that people do not differentiate between the news that are real and the ones that are made-up. The most important channels that facilitate the spreading and increasing of fake news are the ones on which we can connect online and share information with other individuals (Facebook/Twitter/Instagram).

The problem in the online environment is that anyone can post or share something false without being responsible for the consequences of their action. Another problem is when an information goes viral, people just assume that it is real and spread it further resulting in a snowball effect, the number of shares increasing constantly.

Analyzing the reasons of why this happens we've came up with a few ideas. The first one is propaganda; as an example we mention politics, as we are well-aware of the fact that between politicians there will always be place for misinforming communities about their rival. Another reason is the so called fame (one might share fake info about people or events in order to gain attention).

We have also tried to come up with a solution, since we've noticed that this phenomena is getting out of control. We developed an application which lets reader decide if the news he is currently reading is real or not by sharing with them a percentage of how much the algorithm found it to be true and it also gives him the factors taken into consideration by the algorithm.

2 State of the art

The online Cambridge Dictionary defines "fake news" as "false stories that appear to be news, spread on the internet or using other media, usually created to influence political views or as a joke". Thus, there was a need to differentiate between fake or true, which was the starting point in the fake news detection application appearance. There are several software applications available that deal with solving this problem. We will analyze two of them: OIGETIT and Full Fact to understand how it works and what needs to change to combat the trend of fake news.

OIGETIT[9] is an award-winning news app that uses its proprietary AI-powered fake news filtering technology to deliver trusted news to your mobile phone, laptop, desktop, and tablet. The application works well due to its sources(approximately 100.000), every article is processed through their artificial intelligence system based on facts and has assigned a trust rating, however it does not have the ability to filter by region and/or country and most of the users disagree with the percentage of the truth.

Full Fact[2] is building scalable, robust, automated fact checking tools to be used in newsrooms and by fact checkers all over the world. On this site it's presented how they establish if a news it's true or not. The steps made are: Identifying and labelling claims, Matching claims, Taking matching and identifying a step further, Real time checks. Despite the fact that it is free, it is difficult to use because of the registration forms that are required, so the users avoid using it.

Reviewing these examples we can reach some conclusions about why these are not popular: the percentage of the truth is not reliable, so they do not work in all the possible cases; they are not very numerous; they are not easy to use, they rely on simple classification techniques, no inference is done on the text, no ontologies are used.

3 The proposed solution

3.1 Solution Architecture

The proposed solution follows a client-server model and can be split into 3 main components :

- The plugin : the component that is used for user interaction, it collects the data needed for the classification and sends it to the server;
- The text extraction component : processes the data received from the plugin and turns it into parameters that can be used by the classifier component;
- The classifier : receives the required attributes and uses the classification algorithm to tell if the post that needs to be classified can be considered ”news” or not.

The communication between the client and server components is assured by and API. It receives requests from the plugin and calls the text extraction component to parse the request parameters and then passes the output to the classifier.

Figure 1 contains a diagram displaying the principal components of the solution and the general workflow.

3.2 The plugin

The plugin is a Chrome extension consisting of a small software program that customize the browsing experience. It enables users to tailor Chrome functionality

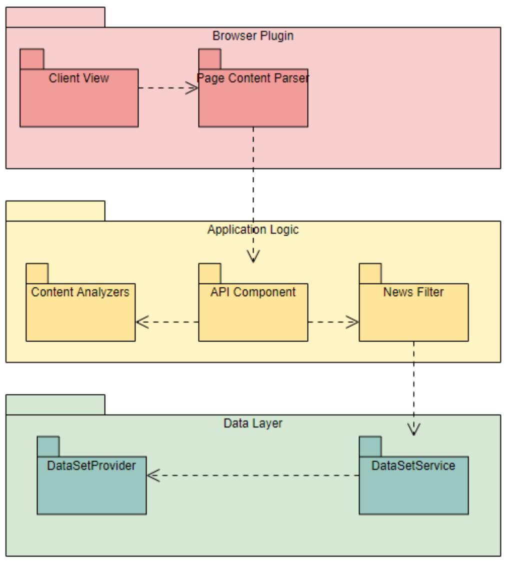

Figure 1: The class diagram of the proposed solution

and behavior to individual needs or preferences which meet the application requirements. It is built on web technologies such as HTML, JavaScript, and CSS. The plugin is the visual component that interacts with the user.

The functionalities offered by the plugin is to identify the posts on the social pages using regular expressions on the url, to extract the source html code and send it to the server, also to communicate with the server related to the analysis of the article and to inform the end user about the accuracy of the post.

3.3 The text extraction component

This module's purpose is to extract the necessary data for the classification, according to the network that is used: Twitter or Facebook. The implementation was based on a whitelist, in order to facilitate the parsing.

To achieve this stage, it is received, via the extension, the URL and HTML of the page from which the post was selected, for the parsing to be performed.

It is known the fact that a public person would not risk their integrity and credibility over posting a fake news, which is why the details related to the user's profile represent a significant criteria in the classification step.

Moreover, this module also contains the language processing algorithm between the maximum number of incorrect words and the total number of words in the text.

3.4 The classifier and the API

The API is the gateway between the plugin, the text extraction component and the classifier. It was developed using Python Flask and only accepts requests using the GET method. Requests using other methods will be rejected. Upon receiving a request, it validates the parameters of the request and, if they are valid, the text extraction module is called, otherwise the request is rejected. After the desired attributes were extracted, they are used as input for the classifier component. The output of the classification is then put in the response and sent back to the client.

The classifier is implemented using a Naive Bayes algorithm. As described in [10], the Naive Bayes algorithm "is one of the simplest approaches to the classification task that is still capable of providing reasonable accuracy". This method of classification is a simple and efficient one that was perfect for the given context. Even though some attributes have some semantic bonds in practice, for example the followers number of the account that owns the post and the number of likes that the post has, these bonds do not exist at a theoretical level, it is possible for someone with 100000 followers to make a post and have 10 likes, even though it is highly unlikely. For this reason, the attributes of a post are considered to be conditionally independent, thus facilitating the use of the Naive Bayes Algorithm.

The dataset used by the classification algorithm is stored in an SQLite database. Since no complex database operations are required, SQLite was chosen as a simple and portable solution. The training data can be found in a table, in contains all the relevant attributes of the social media post along with a credibility index.

The table contains the following columns :

1. Id : the primary key of the table;
2. FollowersNumber : The followers number of the person that the post belongs to
3. Verified : A value that is either 1 or 0 that tells if the account of the post owner is verified or not
4. TweetsNumber : The total number of posts made by that account;
5. Retweets : The number of retweets of the post;
6. QuoteTweets : The number of quote tweets of the post;

7. LikesNumber : The number of likes that the post has;
8. GrammarIndex : A value between 0 and 1 that contains an evaluation of the grammar in the post, 1 meaning perfect grammar;
9. SubjectRelevance : The subject relevance of the post's content, it is a value between one and 100;
10. CredibilityScore : A positive score that is calculated as a weighted sum of the other attributes, the biggest value being 100.

The dataset was filled initially with 10000 automatically generated posts. The classifier does not use the raw values form the table as training data. Instead, a certain range that the value belong to give it's category. There are 4 categories for each attribute, except the verified field, with category 4 being considered the highest and category 1 being considered the lowest. The label is calculated using the credibility index, if it's value is higher than 65, then the label is 1 (the post is considered "news"), otherwise the label is 0 (the post is not considered "news"). After all the values were replaced by their category, the training of the algorithm begins.

This categorisation of the values in the dataset is implemented using and aspect. It intercepts and modifies the output of the method responsable with fetching dataset entries.

All the attributes except the GrammarIndex and the SubjectRelevance are extracted from directly from the post. The other two are calculated using the post's text.

The grammar index is calculated as the division of the correct words number by the the total words number. So if the two numbers are equal, the grammar index will be 1. Otherwise, it will be a value between 1 and 0.

The SubjectRelevance is calculated using a dataset containing 3 million news headlines from the India Express publication. The text is split in individual sentences and then the method that calculates the subject relevance iterates through the dataset. The similarity is for each pair is calculated using the Jaro distance, "a string metric measuring an edit distance between two sequences"[5].

The maximum value of this similarity is kept after iterating through the whole dataset. After the maximum similarity is computed for all the sentences, the subject relevance will be the mean of those values. Since the output of the Jaro distance is a number between 0 and 1, the mean will also be a number between 0 and 1 that is then multiplied by 100 to obtain the subject relevance percentage.

After all the attributes needed for the classification are computed and the algorithm is trained using the dataset, the post will be classified, the possible categories being "news" and "not news". This output is then passed to the API and returned to the client.

4 Results. Evaluation

4.1 Evaluating the plugin

The evaluation of the plugin was done in real-life scenarios. It was installed in the browser and then different websites were accessed. As expected, the websites that are not supported were correctly signaled and no request was made to the API. When browsing on supported sites but not having a social media post focused, the plugin displayed a message that denoted the fact that a post needs to be focused to be classified. Again, no request to the API was made.

4.2 Evaluating the text extraction component

This stage of testing was done by selecting some valid website urls and using them as arguments for this component. The text extraction component was able to correctly retrieve the desired characteristics of the post. Then, some invalid urls were tried, for this stage the component returned null values, an outcome that was expected. No crashes or errors were registered for invalid urls.

4.3 Evaluating the API

To evaluate the API, some stress testing was employed. A Python script was written, it would perform multiple requests to the API with the same parameters, both sequentially and in concurrently. The API only allows GET requests, the test requests that had an invalid method were rejected. The exact number of the requests would be given as a command line parameter, this allowed multiple tests with different requests number.

In the sequential part of the test no notable issues were registered, each request was processed accordingly and the response was the desired one for the given request parameters. The medium latency between the request and the response was approximately 8 seconds, the relatively slow answer is due to the time that it takes for the server to classify the entry.

The API also performed good in the concurrent part of the test, even though it's limitations became visible in the later parts of testing. Due to hardware limitations, the number of parallel was not extremely high. For relatively small numbers, the response time was the as in the sequential part. When the number increased, the latency also increased and eventually requests started getting rejected.

4.4 Evaluating the classifier

The evaluation of the classifier is divided into two parts, it was tested both using real life social media posts and automatically generated ones. In both parts of the test the accuracy was either a little below 80 or between 80 and 86.

First off, automatically generated social media posts were used. To create this posts, all the attributes relevant for the classifier were generated randomly. Even though this led to some improbable cases, for example, a post having 1000000 likes and 100 shares, these situations are theoretically possible. Because of that, all the generated entries were valid ones. The accuracy of this part of the test can be seen below :

Number of entries	Accuracy percentage
250	84.0
500	80.4
750	82.13333333
1000	81.5
100	87.0
200	80.0
300	85.666666
400	85.5

After the first round of testing with automatically generated data, the number of training entries was doubled. This addition of training data improved the accuracy, but improvement was not notable.

Number of entries	Accuracy percentage
100	83.0
200	81.0
300	85.3333333
400	85.5
250	81.2
500	85.8
750	83.866666
1000	84.7
1500	84.666666
2000	84.6
600	83.83333
700	83.42857
800	82.5

For the second part of the test, 100 social media posts were collected from Twitter. These posts belong to political figures, singers, athletes and fan pages of them or other public figures. The accuracy percentage was similar to the one obtained previously. Worth noting is the fact that the number of false positive results were more than four times bigger than the one of false negative results.

Number of entries	Accuracy percentage	False positives	False negatives
100	78	18	4

5 Comparison with other solutions

Compared to other applications, our application approaches the accuracy standards of other existing solutions. As George McIntire describes in [3] an experiment building a fake news classifier using a document-vector model and Naive Bayes approach. He reports an 88 percent accuracy when classifying a fake news dataset which he assembled from various sources. From the point of view of accuracy, our application is also implemented using a Naive Bayes algorithm and has an accuracy of 80 percent.

Another solution with which we compared our program related to accuracy is a solution described in [12] based on Tf-Idf Vectors with Dense Neural Network(DNN). Tf-idf stands for term frequency-inverse document frequency. Tf-idf weight gives an indication on how important a given word is for a sentence or document in relation to the entire corpus. The fully connected dense neural network(DNN) allows to pass the input as sequence of words. The layered architecture allows to experiment with the right depth that is needed for our task. The network consists of an input layer, an output layer and can consist of series of hidden layers. This model takes the Tf-Idf vectors of the headline-article pair, their cosine similarity (standard metric to measure the similarity between 2 non-zero vectors) as input and predicts the output stance, then passed the Tf-Idf vectors to a dense neural network. Tf-Idf on unigrams and bigrams with cosine similarity fed into dense neural network 94.31 percent. The dense neural network represents the words in a hidden dimension which can then be used by the other layers in a seamless fashion. The final dense layer predicts the output probabilities of the stances. Using a finely tuned Tf-IDF Dense neural network (DNN) model, they are able to achieve an accuracy of 94.21 percent on test data.

The number of applications that offer services like our application is not very high, so we are glad that we contribute to the evolution of detecting fake news. Our application is easy to use for users, does not require authentication unlike other existing solutions such as the Full Fact application that requires to complete a form to register, if you want to use their tools and is not easy to access it or use it.

6 Future work

For future work we plan to implement the second part of the detection process, meaning the classification in fake or real news. As we only took the first steps toward an application that can detect news on the user's pages with this accuracy and minimal setup, we plan to continue and improve it further.

Another direction in which we can extend the application is making it available for web pages and other social media platforms. Because the spreading of fake news is happening so fast and is present everywhere on the internet, we will try to make the application available on as many platform as we can.

7 Conclusions

In conclusion, we believe that an application for detecting fake news on social media and on the internet will become even more needed in the future considering the rate of fake news spreading. Since an increased quantity of misleading information can have a bigger impact on our lives that we can imagine, having a tool to help discern the real news is very important.

Also, we believe that the application presented will be well received by its users because of it's intuitive interface and clear purpose. A simple to use and reliable application were our initial goals and we can say we have achieved them.

Bibliography

[1] Jennifer Allen, David Rothschild Baird Howland Markus Mobius, and Duncan J. Watts. "Evaluating the fake news problem at the scale of the information ecosystem". In: (Apr. 2020). DOI: `https://advances.sciencemag.org/content/6/14/eaay3539`.

[2] *Full Fact website*. URL: `https://fullfact.org/`.

[3] *George McIntire-How to Build a "Fake News" Classification Model*. URL: `https://opendatascience.com/how-to-build-a-fake-news-classification-model/`.

[4] Adrian Iftene. "Identificarea știrilor false". In: (Apr. 2017). DOI: `http://itransfer.space/2017/04/20/identificarea-stirilor-false`.

[5] *Jaro–Winkler distance*. URL: `https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance`.

[6] David M. J. Lazer et al. "The science of fake news". In: (Mar. 2018). DOI: `https://science.sciencemag.org/content/359/6380/1094?ijkey=172e1edf77f7a8ef469e1558d99205c04fc42b93&keytype2=tf_ipsecsha`.

[7] Daniel J. Levitin. "Weaponized Lies: How to Think Critically in the Post-truth Era". In: (Sept. 2016). DOI: `https://literariness.org/wp-content/uploads/2019/06/Daniel-J.-Levitin-Weaponized-Lies_-How-to-Think-Critically-in-the-Post-Truth-Era-Dutton-2017.pdf`.

[8] Robert Manea. "Cum putem identifica știrile false? Un scurt ghid practic". In: (June 2019). DOI: `https://mediastandard.ro/cum-putem-identifica-stirile-false-un-scurt-ghid-practic`.

[9] *Oigetit application*. URL: `https://oigetit.com/breaking`.

[10] Mahmoud Parsian. *Data Algorithms Recipes for Scaling Up with Hadoop and Spark*. O'Reilly and Associates, 2015.

[11] Veronica Perez-Rosas and Rada Mihalcea Bennett Kleinberg Alexandra Lefevre. “Automatic Detection of Fake News”. In: (Aug. 2017). DOI: https://www.researchgate.net/publication/319255985_Automatic_Detection_of_Fake_News.

[12] Aswini Thota, Simrat Ahluwalia Priyanka Tilak, and Nibrat Lohia. “Fake News Detection: A Deep Learning Approach”. In: (Apr. 2018). DOI: https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview.