FAKE NEWS DETECTION

The Problem with Fake News:

Fake news is a term used to describe fabricated news stories. Fake news are a falsehood created out of nothing, a lie presented in the guise of true news with the intention of deceiving people. It is important to remember that the information in such news is false, but it appears to be true.

Fake news stories are more widely circulated on social media than real news because they are often more interesting (interesting, but not true). The combination of the speed of dissemination on social networks and the false information contained in fake news is a particularly dangerous combination.

The dangers of fake news:

* Fake news affects our political opinions.
* Fake news causes us to consume less information.
* Fake news leads us to believe in false and dangerous medical information.
* Fake news is dangerous to our lives.



Problem Definition:

The problem that our project aims to solve is the problem of detecting fake news based on the content of the news. The way we approach solving this problem is through the use of machine learning and natural language processing to classify the text of news articles.

Potential Customers:

Applications and social media websites could implement our algorithm to block fake news or warn their users of their presence.

Market Status:

Currently, Facebook is working towards blocking fake news through a system of user reports. Twitter has developed a system that uses machine learning to detect false tweets about the coronavirus.

Project goals:

The goal of this project is to successfully classify real news from fake news based on the text of the news.

Milestones

1. Research the Liar-liar database (we will train our models with this database).
2. Learn relevant natural language processing algorithms.
3. Build an initial prediction model.
4. Feature engineering.
5. Build an improved prediction model.
6. Analyze the results from steps 4 and 5.
7. Repeat steps 4 to 6 until we achieve satisfactory results.

Literature review:

# Fake news detection using machine learning by Simon Lorent

<https://matheo.uliege.be/bitstream/2268.2/8416/1/s134450_fake_news_detection_using_machine_learning.pdf>

Simon Laurent wrote a thesis on fake news classification. In this thesis, Laurent presents various machine learning and natural language processing techniques, using which he built and trained models for fake news classification. Laurent used two datasets - "Fake news" and "Liar-liar". The accuracy rates for classification achieved by Laurent were 94% for "Fake news" and 61% for "Liar-liar".

# Identifying fake news: The LIAR dataset and its limitations by Yingzhao Ouyang

<https://towardsdatascience.com/identifying-fake-news-the-liar-dataset-713eca8af6ac>

In this article, the writer attempted to classify fake news from the Liar-liar dataset using natural language processing techniques. He achieved an accuracy result of 58%.

# ‘Fake news’: Incorrect, but hard to correct. The role of cognitive ability on the impact of false information on social impressions by Jonas De Keersmaecker

<https://www.sciencedirect.com/science/article/abs/pii/S0160289617301617>

This research found that exposure to fake news affects a reader's opinion even after they have been told that the news is false. This discovery suggests that there is a danger to all of us due to the popularity of fake news on social media networks.

Our innovation

There are two main databases for fake news - Fake news and Liar-liar. Fake news contains only fake news posts from social media, and this database has achieved good classification results. In contrast, Liar-liar contains fake news posts from social media, radio, newspapers, and other sources, and as a result, it is considered difficult for machine learning.

So far, good classification results have not been achieved on Liar-liar. We will try to achieve good classification results on Liar-liar because it reflects the fake news in the real world.

Competitor analysis

| Accuracy | Algorithm | Author |
| --- | --- | --- |
| 0.61 | Attention mechanism | Simon Lorent |
| 0.58 | Topic modeling & basic NLP | Yingzhao Ouyang |

The challenge in our project is to achieve a higher accuracy percentage than that of Simon Lorent, who achieved the highest accuracy percentage so far.

Our Solution

We have implemented two different solutions, the difference between the solutions is the way we convert the text in the document to a numerical representation.

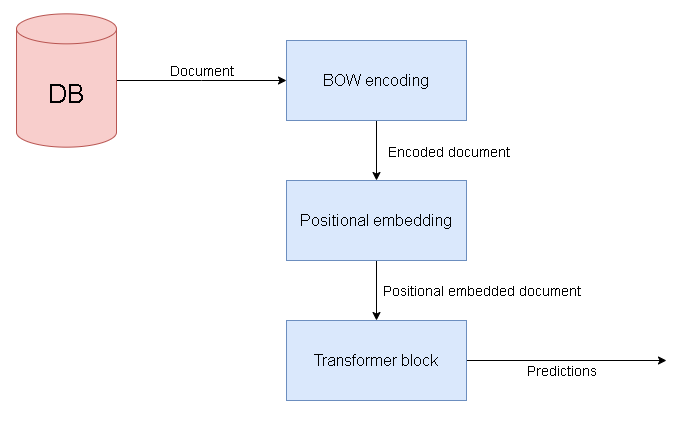
In the first solution, we create a dictionary of words that includes all the words in the corpus, and each word is assigned a unique number. We then convert the words in the document to their corresponding numbers in the dictionary. This method is called Bag-of-words or BOW for short.

In the second solution, we import a trained transformer model called Bert. We feed the documents in the corpus into the model and retrieve the internal representation that Bert learned for each word in the document - an array of vectors.

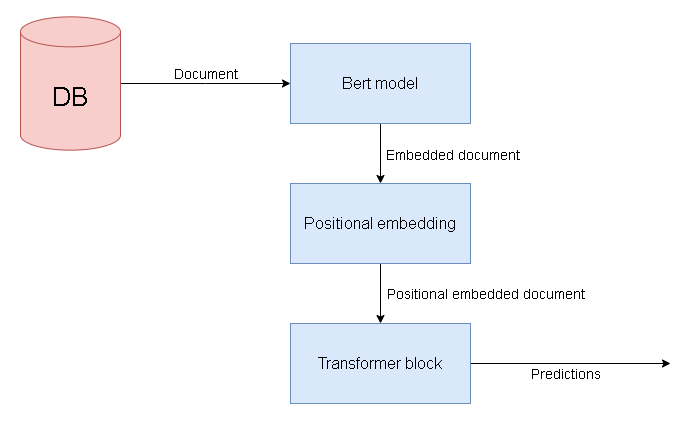
We named the two solutions after the method we used to convert the documents to numerical representation.

Below are diagrams of the solutions:

# Bag of Words



# Bert



Implementation of Bag-of-words

1. Cleaning the training and test corpus:

1.a. Removing special characters and converting all letters to lowercase.

1.b. Removing stop words.

1.c. Stemming each word to its root.

1. Computing TF-IDF values for each word in the corpus.
2. Saving words with high TF-IDF values.
3. Creating a dictionary of the words obtained in step 3, where each word is associated with a unique number.
4. Encoding the documents - each word is replaced with the number it is associated with in the dictionary.
5. Converting the classification labels to One-hot representation.
6. For each document, for each word location in the document, calculating a positional encoding value.
7. For each document, feeding the encoded document to a neuron layer.
8. For each document, feeding the positional encoding values to a neuron layer.
9. For each document, combining the two neuron layers obtained in steps 8 and 9.
10. Building a Transformer model.
11. Training the model:

12a. Splitting the training corpus into batches.

12b. For each batch in the training corpus, for each document in the batch, feeding the model the neuron layer obtained in step 10.

12c. Comparing the model predictions to the classification labels.

12d. Adjusting the model weights using Back Propagation technique.

12e. Repeating steps a to d for a predetermined number of epochs.

1. Evaluating the model against the test corpus.

Implementation of Bert solution

1. Import a trained Transformer model Bert.
2. Feed the corpus to Bert.
3. Obtain the internal representation of each document in the corpus from Bert.
4. Convert the classification labels to One-hot representation.
5. For each document, calculate positional values for all words in the document using a positional embedding formula.
6. For each document, Add positional values to each vector representation of a word.
7. Feed the resulting representation for each document to a layer of neurons.
8. Build a Transformer model.
9. Train the model.

9a. Split the training documents into batches.

9b. For each batch in the training corpus, for each document in the batch, feed the layer obtained in step 7 to the model.

9c. Compare the model's predictions with the classification labels.

9d. Adjust the model's weights using Back Propagation technique.

9e. Repeat steps a-d for a predetermined number of epochs.

1. Evaluate the model on the test corpus.

# The different models

# We built, trained, and evaluated six different models, three based on the BOW approach and three based on the Bert approach. Below is a description of the differences between the models:

| Classification | Labels | Data | Input type | Model |
| --- | --- | --- | --- | --- |
| Multiclass | All train labels | All train documents | BOW encodings | BOW, Model1 |
| Binary | Only ‘true’, ‘false’ labels from train | Train documents labeled as ‘true’, ‘false’ | BOW encodings | BOW, Model2 |
| Binary | All train labels merged to ‘true’ and ‘false’ (\*) | All train documents | BOW encodings | BOW, Model3 |
| Multiclass | All train labels | All train documents | Bert embeddings, M.S.L = 128 (\*\*) | Bert, Model1 |
| Multiclass | All train labels | All train documents | Bert embeddings, M.S.L = 512 (\*\*) | Bert, Model2 |
| Binary | Only ‘true’, ‘false’ labels from train | Train documents labeled as ‘true’, ‘false’ | Bert embeddings, M.S.L = 512 (\*\*) | Bert, Model3 |

(\*) - Merging the labels, we merged the labels "false", "not true" and "barely true" into the label "not true" and the labels "half true", "mostly true" and "true" into the label "true". The goal of merging the labels was to learn binary classification for all the documents in the corpus.

(\*\*) - M.S.L = Max sequence length, when feeding the documents to Bert, this parameter indicates the length of the documents (in words) that Bert receives. We generated two internal representations of the documents, representations that Bert learned for M.S.L = 128 and representations that Bert learned for M.S.L = 512. The goal of the two representations was to check whether documents that are too long constitute noise for the learning process.

Note - It is important to note that Simon Lorent, our main competitor, worked with the same Data and Labels that the BOW, Model2, and Bert, Model 3 models worked with.

Results and Metrics

Database Structure

Our database is divided into training and testing groups. We used two columns from the database - the tags column and the text column.

Tag column - where the classification of the news is specified.

Text column - where the news content is written.

Possible classifications for the news are: False / Mostly False / Half True / Mostly True / True.

All news was collected from an American website called POLITIFACT.COM, which rates the accuracy of statements/news provided by American public figures.

Model Success Evaluation

It is important for us to achieve a high accuracy percentage in order to succeed in classifying the news. In addition, we do not want to make mistakes in classifying fake news (Recall, low FN) (at the expense of classifying true news as fake).

Evaluation Method

After training the model on the training group using the Cross-validation technique, we will calculate the model's predictions on the test group and examine the accuracy and recall percentages.

# Results

| Recall | Accuracy | Model |
| --- | --- | --- |
| 0.013 | 0.293 | BOW, Model1 |
| 0.695 | 0.695 | BOW, Model2 |
| 0.653 | 0.653 | BOW, Model3 |
| 0.000 | 0.201 | Bert, Model1 |
| 0.161 | 0.164 | Bert, Model2 |
| 0.409 | 0.490 | Bert, Model3 |

Summary and conclusions

Results analysis:

Our main measure of project success is to achieve a higher accuracy than Laurent's classification, which received an accuracy of 0.61. We succeeded in the BOW model, Model2, where we received an accuracy percentage of 0.69. The conclusion we draw is that the transformer model is more suitable for fake news classification than the LSTM model used in Laurent's classification.

The results we obtained using Bert's internal representations are low. There are two possible reasons for this outcome:

* Our document lengths are longer than the maximum length Bert can handle (512 words), and the model did not receive enough information (unlike the BOW solution).
* The Bert model is trained for sentiment classification, and its internal representations may not provide quality information for fake news classification like the one-hot encoding we performed in the BOW solution.

Finally, the models fail to perform multiclass classification, with very low accuracy and recall percentages.

Recommendations for further development:

* Try to incorporate more features from the data set.
* The text column can be concatenated with the headline column.
* Use internal representations of documents from other trained models.
* Create and use a sentiment feature of the news item. Sentiment classification can be obtained from a trained model like Hug-face.
* Try different vector representations for word embedding using Bert.

Source list:

Keras transformer model:

<https://keras.io/examples/nlp/text_classification_with_transformer/>

Bert contextualized embeddings:

<https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b>

extracting words with TF-IDF :

<https://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.YIVJ7pAzaUk>