## Abstract

The proliferation of information online serves as a dual-edged sword; while it empowers individuals with access to information, it also exposes societies to the dangers of misinformation and disinformation. With the rapid dissemination of news through social media and other online platforms, the ability to distinguish between real and fake news is crucial. Addressing this issue necessitates the use of diverse machine learning (ML) techniques.

This paper explores the WELFake dataset, it includes 72,134 news articles, with 35,028 real and 37,106 fake news articles. This dataset merges data from four major news sources: Kaggle, McIntire, Reuters, and BuzzFeed Political, to create a more diverse and realistic resource. Our study focuses on three primary areas: the classification accuracy of real versus fake news, the identification of the main characteristic features of fake news, and development of effective methods for summarizing fake news information to enhance public awareness.

The methodology employed includes preprocessing of text, feature extraction using techniques such as Bag of Words, TF-IDF, and Word Embeddings, and the application of machine learning models such as SVM, RNN and BERT for classification. Additionally, clustering algorithms like LDA is used for for topic modeling, and techniques such as Chi square is used to do information extraction and find discriminative words between real and fake news. This research aims to improve media literacy and support the accurate dissemination of information.


## 1- Preprocessing

### 1.1- Text Cleaning and processing

We tried different cleaning methods with the main goal is to reduce dimensionality and retain important words (additional pre-processing steps were done for topic modeling). These steps do not always improve the model performance, that is why we tested and compared them with the raw model. (In the code, the processing steps were mixed on same file). Here are all the steps we tried:

- Cleaning text using regular expressions to retain only alphabetical characters.
- Filtering by frequency (to remove very frequent grammatical words)
- Lowercasing which helps reduce dimensionality
- Removing stop words that do not contribute to meaning
- Stemming which truncates words to their roots and then reduce dimensionality
- POS tagging was used for topic modelling to keep nouns, verbs and adjectives which are the main elements of a sentence that transmits its message or topic.

### 1.2- Feature Extraction

Feature extraction in text preprocessing involves converting textual data into a numerical format that machine learning algorithms can manipulate, typically using methods like Bag of Words, TF-IDF, or Word Embeddings. BoW (Bag of Words) is a straightforward method that counts the number of times each word appears in a document. This representation is a vector where each dimension corresponds to a unique word in the dataset, and the value represents the frequency of that word. BoW is popular for its simplicity and ease of implementation but cannot account for word context.

TF-IDF builds on BoW by reducing the weight of words that appear frequently across many documents, assuming they are less informative. It combines term frequency (TF) with inverse document frequency (IDF),

which adjusts scores based on how common a word is across all documents. While TF-IDF better reflects the importance of terms within documents, it still doesn't capture the context and meaning of words, and its effectiveness may decrease if the document set changes.

Word Embeddings, such as those from Google's Word2Vec, offer a sophisticated approach by embedding words into a vector space where semantically similar words are close. Word2Vec can use models like Continuous Bag of Words (CBOW) or Skip-Gram to predict a word from its context or vice versa. This method captures semantic relationships and reduces dimensionality compared to BoW but requires substantial text data for training and can be computationally intensive.

## 2- Classification

### 2.1- SVM

Support Vector Machines (SVM) are utilized for their robustness in binary classification tasks such as distinguishing between real and fake news. To optimize the performance of the SVM classifier, we employed a hyperparameter optimization strategy using Bayesian optimization. The parameters tuned included the regularization strength parameter ('C', ranging from (1e-6 to 1e+6) on a log-uniform scale), the kernel type (with options including 'linear', 'rbf', 'poly', 'sigmoid'), the kernel coefficient 'gamma' (from (1e-6 to 1e+1) on a log-uniform scale), and the degree (an integer ranging from 1 to 5). This optimization was applied across all representation over 20 iterations with a 5-fold cross-validation. The impact of hyperparameter optimization (HP op) was notably different across the various feature extraction methods:

- Bag of Words (BoW): Initially, SVM utilizing BoW features achieved an accuracy of 0.63. After employing hyperparameter optimization, the accuracy notably improved to 0.89.
- TF-IDF: The accuracy started at a relatively high 0.87. Post optimization, the model's accuracy was slightly boosted to 0.91. However, this improvement came at the cost of high computational resources, suggesting that the gains might not justify the extensive computational effort required for this method.
- Word Embeddings: Starting with an initial accuracy of 0.83, the application of Bayesian optimization led to a slight increase in performance, reaching 0.85. This representation was the lowest performing, indicating that semantics may not necessarily be relevant for news classification.

### 2.2- RNN

Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) units are well-suited for processing sequenced data, making them ideal for text analysis where context and word order are important. For this project, we employed word embeddings as inputs to the LSTM model, enabling the network to leverage the semantic relationships encoded within these embeddings.

LSTMs effectively address the limitations of traditional RNNs, such as the vanishing gradient problem using gates. These gates regulate the flow of information by deciding what to retain and what to discard, thus allowing the model to maintain relevant context over extended text sequences.

The LSTM model reached an accuracy of 0.85, demonstrating the advantages of Recurrent Neural Networks (RNNs) for text processing. This performance surpasses that of the SVM using the same word embeddings representation, underscoring the RNN's capability to manage sequential data effectively. Due to the computational demands, Bayesian optimization was impractical for this setup; thus, grid search was employed to fine-tune the model parameters. The most effective configuration was achieved with the initial settings, indicating that the starting parameters were well-suited for capturing the complexities of the dataset.

### 2.3- Transformers

BERT, or Bidirectional Encoder Representations from Transformers, stands out as a powerful tool for various Natural Language Processing (NLP) tasks, including text classification for identifying fake news. Its effectiveness lies in its unique architecture and training approach.

At the core of BERT lies a multi-layered Transformer encoder. This powerful architecture allows BERT to process text in a way that older methods cannot. By considering each word's context from both the preceding and following words, BERT achieves a bidirectional analysis. By comprehending the word's position within a sentence, this leads to a much richer interpretation of the text's overall meaning.

The advantage of BERT tokenizer relative to Word2Vec is that it can distinguish different senses of a word with its contextual embeddings. In fact, Word2Vec struggles with words that have multiple meanings or words that share the same spelling but have different meanings. Another advantage is that BERT, with its self-attention mechanism, can capture bidirectional context from the entire input sequence. This enables BERT to model long-range dependencies and capture relationships between distant words more effectively than Word2Vec. Plus, BERT is pre-trained on a large corpus of text from Toronto BookCorpus (800M words) and English Wikipedia (2,500M words) which allows it to capture rich linguistic patterns and semantic relationships from diverse text sources. Comparatively, Word2Vec takes a simpler approach compared to BERT's pre-trained tokenizer. It typically trains on a smaller dataset of text using shallow neural networks like CBOW or Skip-gram. These models work by predicting surrounding words based on a target word, or vice versa, within a limited window of surrounding words. While Word2Vec can identify some semantic connections and similarities between words, its embeddings don't account for the broader context. This limitation hinders its ability to capture the full complexity and subtle nuances of language.

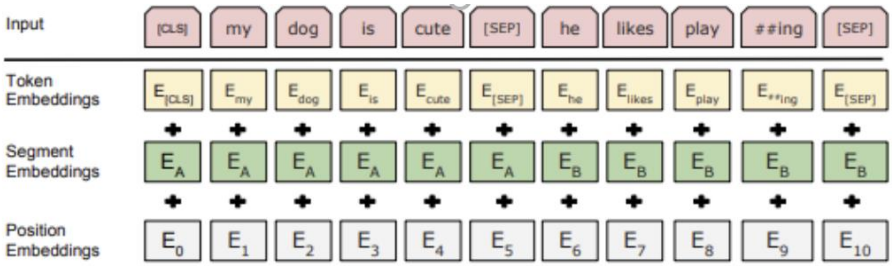


*Figure 1: Bert input representation.*

First for the BERT classification to classify fake from real news, the titles and texts from news articles must be transformed into numerical data, and this numerical format is represented vectors. These vectors, also called embeddings capture the meaning of the text and are crucial for training the BERT model. Different from previous traditional classification models seen, BERT doesn't require preprocessing the data. Due to the nature of BERT's architecture and pretraining objectives, the model can handle raw textual data directly without the need for extensive preprocessing. This simplifies the preprocessing pipeline and allows BERT to capture complex linguistic patterns and semantic relationships more accurately with the following embeddings:

Token embeddings are the foundation upon which BERT understands the meaning of text. Each word or sub-word unit in the text is mapped to a corresponding vector in a pre-trained embedding matrix. This vector representation captures the word's meaning and its connections to other words within the vocabulary. The embedding matrix itself is not created by BERT, but rather learned during a massive pre-training stage on a vast amount of text data. By referencing this pre-trained matrix, BERT assigns a meaningful vector to each token, allowing it to understand the text it is analyzing. Additionally, token embeddings can also capture sub-word units in texts. This is crucial for identifying fake news that often employs misleading or manipulative wording due to BERT being able to discern subtle differences in languages.

In BERT sequences, sentences usually are paired and packed together into one single sequence. To differentiate between the two sentences, segment embedding is utilized for this matter. It's represented as a binary representation, 0 represents the first sentence and 1 represents the second. By assigning such a code to each token's embedding, BERT can differentiate between the meaning of words depending on which segments it

belongs to. This is particularly useful in detecting fake news that may contain contradictory or inconsistent information across different segments or paragraphs.

BERT, being a transformer-based model, does not inherently understand the order of tokens in the input sequence. To fill in this information, position embeddings are introduced. These embeddings encode the relative or absolute position of each token within the sequence. By incorporating these positional cues, BERT can identify how the meaning of a word can change depending on its position in the sentence. In the case of fake news detection, by understanding word order, BERT can analyze the flow of events or arguments in an article and pinpoint inconsistencies that might be indicative of fakery. For instance, BERT can identify articles where the chronology of events seems illogical or where the text includes contradictory statements. Token, segment and positional embedding are then added together to form the inputs into the model.

Input IDs are then used to retrieve the corresponding embeddings from the pre-trained embedding matrices. When processing an input sequence, the model searches up and finds the embedding vectors associated with each input ID in its embedding matrices. These embeddings represent the semantic content of the tokens in the input text and serve as the input features for the model. Therefore, each token in the text is replaced by its respective input ID.

BERT models have a maximum input length they can handle. If the length of the input text exceeds this limit, truncation is necessary. Truncation involves removing tokens to ensure it fits within the maximum input length allowed by the model. Truncation helps maintain computational efficiency and prevents memory issues during model training and inference.

Padding is the process of adding special tokens (usually zeros) to the shorter sequences within a batch to match the length of the longest sequence. This ensures uniformity in input size and enables efficient batch processing. Padding tokens carry no meaningful information and are ignored by the model during computation.

Attention masks are binary masks that indicate which tokens are actual words and which ones are padding tokens. Since padding tokens carry no useful information, attention masks are used to instruct the BERT model to pay attention only to the actual tokens while disregarding the padding tokens during computation. By applying attention masks, the BERT model can focus solely on the relevant tokens while processing the input text.

We utilize the pre-trained BERT model architecture for sequence classification, specifically the 'bert-base-uncased' variant. This variant is trained on uncased English text and consists of 12 transformer layers. We initialize the model for sequence classification by specifying 2 as our number of output labels. The model is loaded onto the device GPU from google collab to utilize paralleled computation power.

By inputting the input_ids and attention masks into Py torch, the model is then trained using the AdamW optimizer, which is a variant of the Adam optimizer with weight decay. We employ a learning rate scheduler of 5e-5 to adjust the learning rate during training, which can help improve convergence. The training process is carried out over 6 epochs. During each epoch, the model is trained on batches (64) of data from the training set. Gradient clipping is applied to prevent exploding gradients, and the optimizer updates the model parameters based on the computed loss.

After each epoch, the model's performance is evaluated on the validation set. This evaluation involves computing the loss and accuracy of the model predictions. The loss indicates how well the model is performing in terms of minimizing prediction errors, while accuracy measures the proportion of correctly classified instances. By monitoring both metrics during training, we can assess the model's progress and identify any overfitting or underfitting issues. In our case, we did not see overfitting since both the training and validation accuracies are high and close to each other.

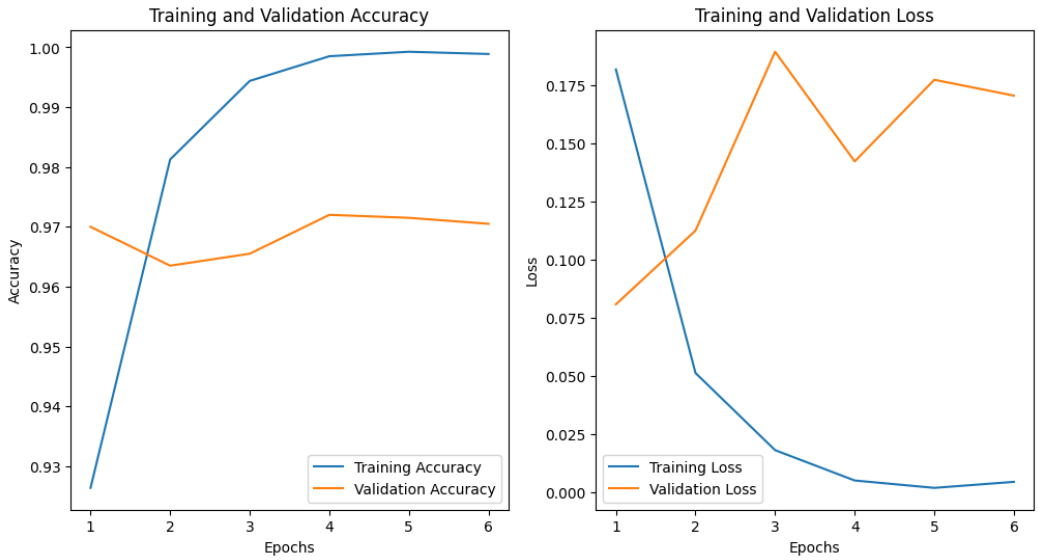


*Figure 2: Training and Validation accuracy and loss*

After processing all validation data, we identify tokens with the highest average attention weights. This is accomplished by sorting the tokens based on their average attention weights in descending order and selecting the top 20 tokens with the highest attention weights. These tokens are considered the most influential and discriminative in the classification of fake news. However, the top 20 tokens found are not indicative of which words are predominantly associated with either fake or real news. Plus, they are too generic to give significant meaning. The tokens are: ["[SEP]", "[CLS]", "title", ":", ";", "text", "the", ",", ".", "to", "a", "of", "-", "s", "in", "on", "and", "(", ")", "|"]

We also utilized TextRank for text summarization, it condenses lengthy news articles into shorter versions that retain their core meaning. Extractive summarization from TextRank, identifies important sentences through features like word frequency and semantic similarity. These sentences are then ranked, and the top ones chosen to form the summary. By incorporating summarized text into BERT models, we expected to be able to train the BERT models at a faster speed without compromising the essential elements from the news articles

**2.5 Results**

Table 1: Performance result of the classification models used

| Model | Accuracy (%) | Precision (%) | Recall(%) | F1-Score(%) |
|---|---|---|---|---|
| All fake news | 50.53 | 50.53 | 100.00 | 67.15 |
| SVM (BOW) | 88.91 | 84.87 | 94.99 | 89.65 |
| SVM (TFIDF) | 90.76 | 90.58 | 90.67 | 90.58 |
| SVM (WE) | 85.11 | 85.11 | 85.17 | 85.25 |
| RNN (LSTM) | 85.11 | 85.34 | 85.17 | 85.26 |
| BERT | 96.70 | 96.92 | 96.73 | 96.82 |
| BERT TextRank | 96.55 | 97.00 | 96.35 | 96.67 |

The table above shows that the BERT classification model outperforms all others. BERT captures context from both the left and the right sides of a token within a sentence, unlike traditional models that only read

sequentially from left to right. This capability makes BERT especially effective in understanding the nuances and complexities of language, leading to more accurate predictions in text classification tasks.

For the BERT model that utilized text summarization, the training time was not faster than the model utilizing raw text, but the performance results were on par with the raw BERT model. It performed 1% lower on the recall score. Therefore, TextRank was able to correctly extract influential sentences that can identify fake news.

## 3- Topic modeling & information extraction

Our main goal was to do clustering with the help of some unsupervised algorithms and methods seen in class and then do feature extraction within the clusters found to derive the main themes and topics for each cluster, hence discovering the main topics covered in fake news. We found that the methods we saw are not very suitable for large datasets as they are based on distance calculations and performance deteriorates as well. Secondly, some of them, like k-means, are hard clustering and may not consider that some documents are a mixture of different topics. We thought using word embeddings with gaussian mixture but found a better algorithm called LDA.

LDA topics are straightforward to interpret because they are represented as distributions of words, allowing each topic to be easily described by its most probable words. This aligns well with how humans categorize textual themes, making the topics intuitive and directly readable. In contrast, GMM clustering applied to word embeddings deals with high-dimensional, continuous vector spaces where semantic relationships are captured. These clusters can include a mix of semantically related but contextually diverse words, making them less immediately obvious and harder to interpret. Because of this complexity and abstractness in relationships, understanding and describing GMM clusters requires deeper analysis, unlike the simpler, word-based interpretation provided by LDA.
The LDA algorithm has 3 hyperparameters which we tried to manually optimize for a range of values for each parameter.
The 3 hyperparameters were:

- K = number of topics(pre-defined)
- Alpha: Alpha is a hyperparameter that controls the sparsity of the document-topic distribution. A low alpha value means that each document is likely to be composed of a small number of topics, while a high alpha value means that documents are likely to contain a more diverse mixture of topics
- BETA: Beta is a hyperparameter that controls the sparsity of the topic-word distribution. A low beta value means that each topic is likely to be composed of a small number of words, while a high beta value means that topics are likely to contain a more diverse set of words.

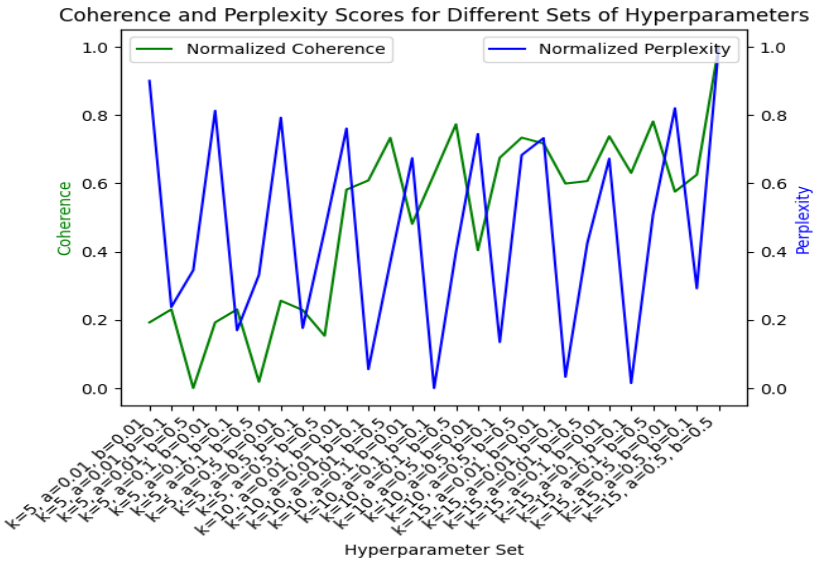


*Figure 3: Coherence and perplexity scores for different sets of hyperparameters*

Best Hyperparameters:

| | num_topics | alpha | beta | coherence | perplexity | total_score |
|---|---|---|---|---|---|---|
| 13 | 10 | 0.1 | 0.1 | 0.626347 | 0.0 | 0.626347 |

*Figure 3.1: Highest combined score (highest coherence, lowest perplexity)*

We went eventually with the highest coherence score (ignoring the highest combined score of perplexity and coherence in figure 3.1 because we got more similar words semantically, per topic, considering only highest coherence) .We got the highest coherence (measuring best similarity between words per topic with higher parameters) with K=15, beta=0.5 and alpha=0.5 (see figure 3 above). We got the results below : the topics are centered about : illegal immigration, national security, sports, climate change, law and legislation etc ... we were also able to identify specific topics with the keywords found, like the assassination of the king of Sweden ( topic 8), Trump election campaign ( Topic 6), Trump imposing nuclear sanctions on China ( topic 12), the british brexit ( topic 3) , Angela Merkel German elections ( Topic 5), etc ... again with better hyperparameter tuning ( like grid search which we did not have GPU resources to do ) we could have obtained better results.

```
Topic 0:
authoritarians authoritarianism hill turnbull boehner authoritarian runners electric spillway flower
Topic 1:
government trump russian officials president campaign investigation office security election
Topic 2:
military government state security islamic forces killed syrian group attack
Topic 3:
brexit gun british time new hotel guns leave city flight
Topic 4:
police school students family officers man city time black children
Topic 5:
coalition merkel party parliament new german parties labour election government
Topic 6:
trump campaign party election presidential president voters republican political democratic
Topic 7:
game team games players season sports time new league athletes
Topic 8:
vitamin gang king sweden swedish coulter palace molins church murders
Topic 9:
water climate storm hurricane change warming fish scientists carbon power
Topic 10:
tax law government state federal health new court house legislation
Topic 11:
women time new life work world good company woman film
Topic 12:
nuclear trump deal trade sanctions countries agreement world chinese new
Topic 13:
kurdish independence iraqi referendum government region kurds catalan oil spanish
Topic 14:
immigration order border immigrants workers illegal ban executive jobs country
```

*Figure 4: Words associated with the 15 topics found using LDA*

**Information extraction:**

Lastly, we tried to do information extraction to recognize the main words that are mostly associated or differentiate the two classes of fake news and real news. We used TF-IDF representation along with chi-square method. The chi-square test assesses whether observed categorical data deviates significantly from what is expected under the hypothesis of no association between classes. To apply it, we create a contingency table listing observed frequencies of categories (like attribute levels) across two classes. We calculate expected frequencies for each table cell assuming no class-attribute relationship. The chi-square statistic is then computed by summing the squared differences between observed and expected frequencies, each divided by the expected frequency. A significant test result indicates that the attribute significantly discriminates between the two classes.

You can see the results from the figure 6 below (we filtered some useless words that appeared in results to get better meaningful words with the next best chi-square score).

From the graph below, we interpreted the results as follow : the real news are based on evidence or online platforms that can be investigated ( either images, videos, features, Facebook, twitter etc. ...) or containing words identifying the fake news ( propaganda, fake etc ...) , or affirming the facts ( truth, true, fact, real), other words may have specific topics associated to them which are hard to interpret with just words without context. On the other hand, the highest chi-square score was for Reuters (a reliable news agency) which led us to believe that a lot of fake news associate themselves with real news agency to try to legitimize their fake news and make them believable. The other words are very similar to the ones found in topic modeling and touch upon national security, politics, law and legislation etc... Here is the full graph below:

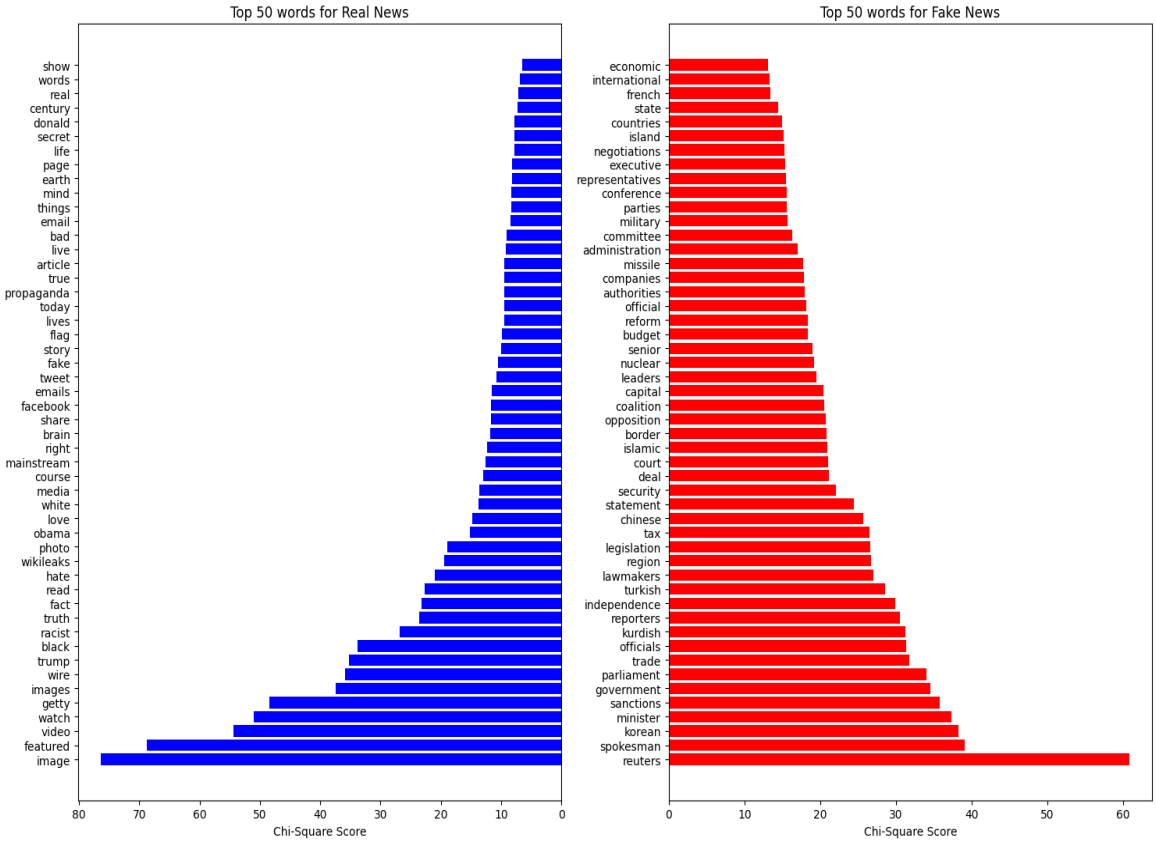


*Figure 6: Words that mostly represent the 2 classes (fake and real news)*

## Conclusion

In this study, we began our analysis by employing classification techniques to distinguish between real and fake news, using models such as SVM, RNN, and BERT. While each model provided good results, the BERT model showed a substantially better accuracy performance since it captures the contextual nuances of the text, making it particularly adept at identifying misinformation in the news.

Following classification, we explored clustering to uncover underlying themes and patterns within the data. Employing Latent Dirichlet Allocation (LDA), we successfully identified dominant topics such as national security and politics, which frequently appear in fake news narratives. A lot of fake news people associate themselves with reliable news agencies to try to legitimize their stories. One can notice real news if more concrete evidence is put on the table backing up the news (images, videos, etc.). We discovered interesting method to gather topics and keywords related to fake news and real news using chi-square method. Our optimized LDA settings, which included specific topic numbers and finely tuned hyperparameters, resulted in high coherence scores, underscoring the clarity and relevance of the extracted topics.

## Sources

Smith et al. (2023) on *linguistic patterns in fake news* extracted from: https://www.researchgate.net/publication/350665094_WELFake_Word_Embedding_over_Linguistic_Features_for_Fake_News_Detection

Data extracted from Kaggle: https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification/data?fbclid=IwAR2EQSEoeT6xXI79KhSzO3rxDnGHrQgsDmXcg-b22AeE8-krQ2jlmVoxfIA