# Fake News Detection

**Apostol Alin-Constantin**
alin-constantin.apostol@s.unibuc.ro

**Ciuperceanu Vlad-Mihai**
vlad-mihai.ciuperceanu@s.unibuc.ro

**Pîrvulescu Daria-Maria**
daria-maria.pirvulescu@s.unibuc.ro

## Abstract

This paper aims to highlight the importance of recognizing synthetic online content through the use of Natural Language Processing. In today's society, disinformation can range from images and videos altered using AI tools to entirely fabricated news articles and election posts. That being said, by leveraging NLP techniques, we can develop automated systems capable of identifying patterns of synthetic content. These systems can be used to flag potential disinformation, assist fact-checkers, and increase public awareness of deceptive content circulating online. We employed a range of models, including traditional machine learning approaches such as Support Vector Machines (SVM) and Random Forests, as well as state-of-the-art techniques based on Transformer architectures. These models were utilized both for binary classification of news content into fake or real and for identifying and extracting the fabricated segments within the text.

## 1 Introduction

Synthetic media can cause significant harm across various domains such as elections (US election from 2020), healthcare (Covid pandemic vaccine misinformation), and public safety. We think that is important to acknowledge this problem and seek competent solutions to prevent disinformation of the mass.

## 2 Related Work

The domain of fake news detection has experienced rapid growth in recent years, driven by the increasing prevalence and impact of fake news. As a result, a diverse body of research has emerged, exploring various methodologies ranging from traditional machine learning approaches to state-of-the-art large language models. We list some important or interesting discoveries from latest to most recent:

1. *Fake news detection on social media: A data mining perspective*(Shu et al., 2017) where a comprehensive survey was firstly done on this topic, followed by classic ml approaches to this task (Logistic Regression, SVM, Random Forest etc);

2. *Proppy: A system to unmask propaganda in online news* (Barrón-Cedeno et al., 2019) presents a system designed to detect propaganda in news articles, with the aim of promoting media literacy and helping readers distinguish between factual reporting and persuasive or manipulative content;

3. *FakeBERT: Fake news detection in social media with a BERT-based deep learning approach* (Kaliyar et al., 2021) where an improved *BERT* was used to classify fake news from real news. The results where impressive:

   > *'Our proposed model (FakeBERT) produced more accurate results as compared to existing benchmarks with an accuracy of 98.90 %.'* (Kaliyar et al., 2021)

4. *WELFake: Word embedding over linguistic features for fake news detection* (Verma et al., 2021) where the model WELFake is introduced, making use of word embeddings and having a promising accuracy and technique ;

5. *AI-assisted deep NLP-based approach for prediction of fake news from social media users* (Devarajan et al., 2023) which proposes a model that detects fake news with over 98% accuracy over different datasets.

## 3 Method

In this section, we present the methods employed to address the problem of propaganda detection in online news. Our approach includes both traditional machine learning and state-of-the-art deep learning techniques. Specifically, we experimented with

classic classifiers such as Support Vector Machines (SVM) and Random Forests, using Word2Vec embeddings (both Skip-gram and CBOW) trained from scratch as well as pretrained models. Additionally, we evaluated several Transformer-based architectures, including BERT(as introduced by J. Devlin (Devlin et al., 2019)) and its variants such as DistilBERT and TinyBERT. To better understand the dataset, we conducted exploratory data analysis using word clouds and bigram frequency visualizations. Furthermore, we performed token-level analysis to identify which parts of the text are indicative of fake or propagandist content.

## 3.1 Dataset

### 3.1.1 Description

The dataset used in our experiments was obtained from Kaggle (Shahane, 2021) and originates from the study conducted by (Verma et al., 2021).

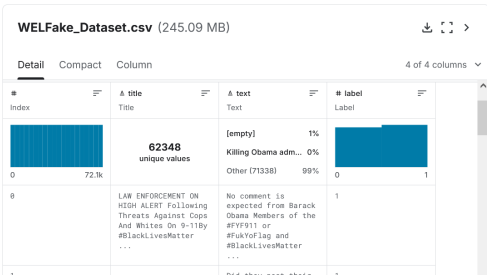



Figure 1: Dataset View

The dataset has 4 columns:

1. index which keeps track of how many entries are in the CVS (in the raw data there are 78098 rows which gives plenty of data for training our models);
2. title (the title of the news article);
3. text (the text of the news article);
4. label (0 for fake news and 1 for real news, so the data was automatically annotated).

### 3.1.2 Issues and Relevance

The dataset we selected was relatively large but exhibited several imperfections, including empty strings, non-English texts, hyperlinks, and emojis. Consequently, we performed several preprocessing steps to clean the dataset. First, we removed all entries that were not valid textual strings, such as emojis, null values, and hyperlinks. Next, we filtered the dataset to retain only English-language content. These steps were necessary because non-textual entries cannot be meaningfully classified as fake or real news, and non-English text could introduce noise and negatively impact the performance of our models during training. We used the *langdetect* library to automatically detect the language of our input.

```
Language distribution in original dataset: {'en': 70703, 'fr': 43, 'ar': 19, 'de': 110, 'es
': 145, 'ru': 156, 'sw': 14, 'pl': 4, 'so': 4, 'pt': 12, 'it': 6, 'tr': 7, 'nl': 5, 'fi': 3
, 'tl': 3, 'hr': 3, 'et': 1, 'vi': 2, 'ro': 4, 'no': 5, 'da': 1, 'af': 3, 'el': 2, 'zh-cn':
1, 'hu': 1, 'cy': 1, 'ca': 1, 'sq': 1, 'id': 2, 'sv': 1, 'lt': 1}
Empty texts removed: 783
Non-English texts removed: 561
Cleaned dataset saved to 'dataset_clean.csv'
```

Figure 2: Dataset Cleanup

### 3.1.3 Statistics

To gain a clearer understanding of the data distribution and the linguistic characteristics associated with fake news in the dataset, we generated word clouds and bigram visualizations.

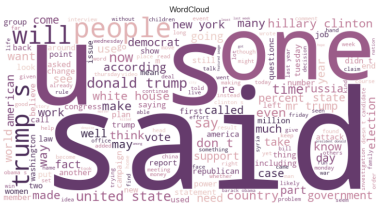



Figure 3: WordCloud

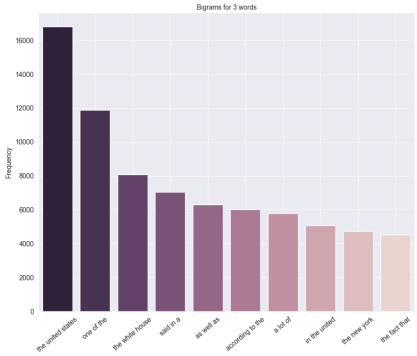



Figure 4: Bigram for the most frequent 3 word groups

As seen in the graphics, country names and president names are the most frequent and we expected it to affect the training of the model.

Nonetheless, we combined the *text* and *title* columns into a single input and lowered all the letters. The dataset was split into three categories: *train (56142 rows), validation (7018 rows)* and *test (7018 rows)*.

### 3.2 Preprocessing

#### 3.2.1 Overall

The current input for training is formed from CSV containing the the text and the labels. To make it usable for a NLP task, we used different embeddings: **WordToVec-CBow, WordToVec-SkipGram** and **Glove** both pretrained and trained from scratch. We used more embeddings to see if it affects the overall accuracy and data distribution in the fake news category. The results will be presented with the models used in this section: 3.3.

#### 3.2.2 Brief description

**Word2Vec – CBOW (Continuous Bag of Words)**:

CBOW predicts a target word based on its surrounding context words. Given a window of words before and after a missing word, the model learns to infer the most probable central word. It's efficient for smaller datasets and works well for frequent words.

**Word2Vec – Skip-gram**: Skip-gram works in the opposite way of CBOW: it predicts the surrounding context words given a single target word. It is especially effective at learning representations for rare words and performs better on larger datasets.

**GloVe (Global Vectors for Word Representation)**:

GloVe builds word embeddings by factorizing a global word co-occurrence matrix. It captures statistical information about how often words co-occur in a corpus, combining the advantages of both global matrix factorization (like LSA) and local context-based learning (like Word2Vec), resulting in meaningful vector representations.

### 3.3 Method

All code, plots, and detailed analyses are available in our public GitHub repository (Apostol Alin-Constantin, 2025). In the following sections, we present the core methodology and highlight the most interesting results.

#### 3.3.1 Classic ML

**Implementation**

We used two classic machine learning models for this part: Linear SVM and Random Forest, pairing each with three word embeddings: Word2Vec-CBOW, Word2Vec-SkipGram, and GloVe (both in their pretrained form and trained from scratch). This resulted in eight combinations that emphasize the importance of selecting the appropriate embedding for a given dataset. While model accuracy is not the primary focus of this research, all combinations achieved over $84\%$ accuracy, demonstrating that classical machine learning can still be a viable solution for this problem.

**Comparison**

Comparing our implementation with a well-known article from 2017 (Shu et al., 2017), we can say that the overall accuracy has improved for these models over time (the best baseline is a $82\%$ accuracy). Even though state-of-the-art techniques like Transformers perform much better at this task, it is important to remember that these models offer practical advantages: they are easier to use, more efficient in terms of time and memory, and generally easier to interpret.

As for comparing our two models the best combinations were: Random-Forest with Word2Vec-SkipGram trained from scratch with $92,49\%$ accuracy with Linear SVM Word2Vec-SkipGram trained from scratch with $94,54\%$ accuracy. As described in here 3.2.2, it seems that Skip-Gram worked best for our large dataset.

```
word2vec_model = Word2Vec(
    sentences=X_train_tokens,
    vector_size=100,
    window=5,
    min_count=1,
    workers=4,
    sg=1,
    seed=42
)

svm = SVC(
    kernel='linear',
    probability=True,
    random_state=42
)
```

Listing 1: Configuration of the best models

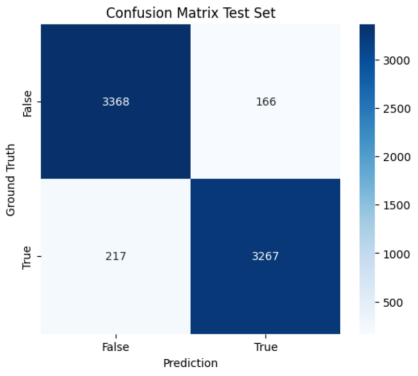



Figure 5: SVM Confusion Matrix

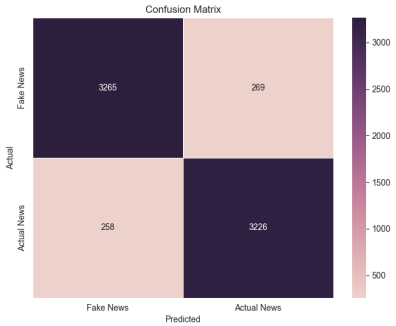



Figure 6: Random Forest Confusion Matrix

Another interesting analysis is to see which words are the most frequent in the fake data predictions made by those models. For this task we created WordClouds and bigrams that are available here: (Apostol Alin-Constantin, 2025).

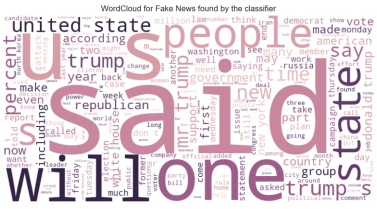



Figure 7: WordCloud for fake news category

### 3.4 Transformers

**Introduction**

Transformer-based models have revolutionized the field of Natural Language Processing by introducing self-attention mechanisms that allow the model to capture long-range dependencies and contextual information more effectively than previous architectures. Currently, these types of models have led to foundation models and the state-of-the-art approach for all kinds of tasks, providing superior generalization capabilities.

**Architectures**

For our fake news detection task, we evaluated the performance of three popular Transformer architectures: BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), DistilBERT, and TinyBERT. These models achieved close to perfect results in classification, while also showcasing a trade-off between accuracy and computational efficiency.

While BERT is the standard, popular model, its distilled versions came close in metrics, being lighter alternatives optimized for resource-constrained environments.

**Use of pretrained models**

For a better model capacity, we used the pretrained versions of each of them, fine-tuning them on our dataset. Thus, we can use the advantage of having a model trained on a large dataset that learned complex language correlations for our specific detection task.

**Tokenization**

In order to fine-tune the models, the text was processed using the corresponding tokenizer for each model (e.g. BertTokenizer), furtherly building the dataloaders with the token ids, attention masks and news labels.

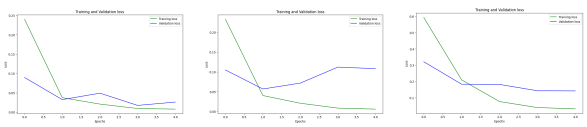
To ensure a uniform input format, we set the maximum length of tokens to 512, enabling truncation and padding to cover all cases.

Due to working with a dataset of significant size, training a large model such as BERT on the whole data was challenging time-wise. Therefore, we randomly chose a subset of 4000 subsamples for training and 500 for validation, keeping the same ratio as in the WELFake dataset.

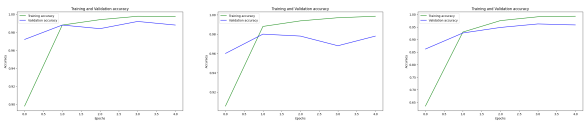
**Fine-tuning**

The models were fine-tuned for 5 epochs, using an AdamW optimizer with a learning rate of 1e-5 for the two bigger models, respectively 1e-4 for TinyBERT and the models' internal loss.

This short fine-tuning proved to be extremely effective, with an immediate increase in validation accuracy and a sharp decrease in the train loss. The amount of time it took corresponded to each level of complexity: 33 minutes for BERT, 17 for DistilBERT and only 3 for TinyBERT (which puts into perspective the aggresive compression technique used for this version). Below can be seen the plots of the training metrics - loss and accuracy:



(a) BERT (b) DistilBERT (c) TinyBERT

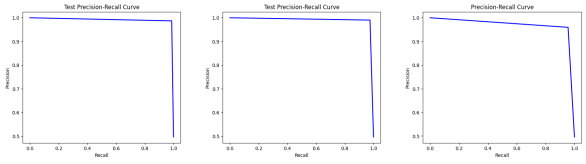


(d) BERT (e) DistilBERT (f) TinyBERT

**Results Analysis**

All models achieved exceptional results, with **BERT** achieving the highest accuracy of **98.72%**, closely followed by **DistilBERT (98.30%)** and **TinyBERT (95.87%)**.

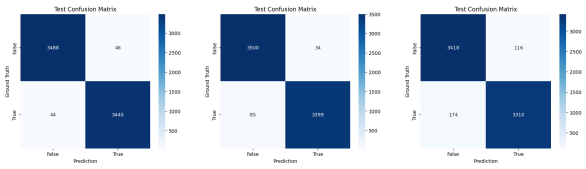
The precision-recall curves below confirm these results:




(a) BERT (b) DistilBERT (c) TinyBERT

Interestingly enough, it can be noted from the validation accuracy plots that the pretrained BERT model was capable of obtaining a validation accuracy of 97.20% after only one epoch, showing just how deep the model understands general language patterns that can be translated to any task. The trade-off between results and model size can be noticed for TinyBERT, being a few percents behind, only slightly above SVM. Still, it is the fastest method with such a high accuracy.

The confusion matrices below depict the small number of misclassified news, having only 90 such examples for BERT:




(a) BERT F1-score: 0.987 (b) DistilBERT F1-score: 0.982 (c) TinyBERT F1-score: 0.958

**Insights**

While BERT provides the best result and TinyBERT provides a strong result in the shortest time period, DistilBERT proves to be the best trade-off of the BERT versions. Fine-tuned in half of BERT's time, it produces an accuracy only 0.52% lower, establishing itself as a reliable model for further use.

These types of trade-offs are important, opting for TinyBERT for real-time applications, DistilBERT for short-time experiments and BERT for the best possible metrics.

## 3.5 Fake part extraction

Beyond classifying entire news articles as real or fake, a deeper layer of interpretability can be achieved by identifying which specific tokens contribute most to the model's decision. This level of analysis is crucial in understanding potential patterns of propaganda or disinformation, as well as the internal decision process of the model.

To this end, we performed token-level attribution analysis using two popular interpretability techniques: LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016) and Integrated Gradients from the Captum library (Sundararajan et al., 2017).

These tools allow us to visualize the most influential words for a given prediction and assess whether the model relies on semantically meaningful cues or artifacts in the data. The goal was not only to validate model reasoning but also to provide transparency in high-stakes domains like fake news detection.

### 3.5.1 LIME

**Introduction**

LIME is an interpretability method designed to explain the predictions of any classifier in a human-understandable way by approximating the model locally with an interpretable surrogate model.

To apply LIME in the context of NLP, a specific text instance is selected, and LIME perturbs this text by removing or altering words to create a set of similar samples. These perturbed texts are then passed through the original model to observe how its predictions change (more specifically, a function that returns the probabilities is passed to LIME). Based on the outputs, LIME trains a simple, interpretable model, typically a linear model, on this synthetic dataset, weighted by how close the perturbed samples are to the original instance.

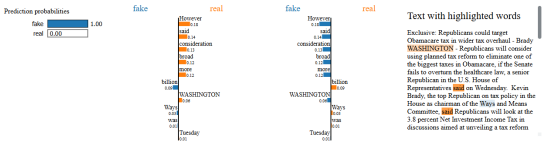
The result is a set of weights assigned to individual tokens, indicating how much each word contributed positively or negatively to the model's decision.

As we mentioned earlier, LIME is model-agnostic, meaning it does not require access to the internals of the model (like gradients or attention), which makes it especially useful when dealing with black-box models or when model internals are unavailable.

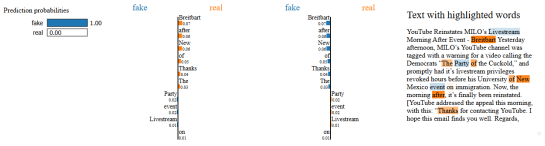
**Results Analysis**

We used the previously fine-tuned DistilBERT model for predicting the probabilities, setting it to look after the first 10 most important words and 100 perturbated text variations to be created in order to determine how the probabilities change.

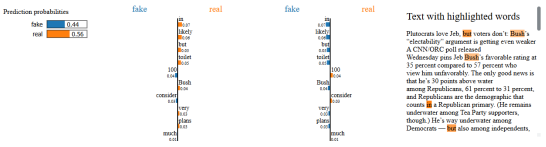
We chose 2 correctly identified news and 2 misclassified ones:



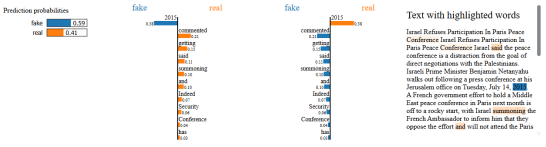
(a) Correctly classified 1



(b) Correctly classified 2



(c) Misclassified 1



(d) Misclassified 2

It seems to be sensitive to important named entities (e.g. "Bush", "Washington"), high numbers (e.g. "100", "billion"), potential fake statements spoken (e.g. "said", "commented"), important events where lots of people attend/watch(e.g. "conference", "livestream"), adverbs or conjunctions used for strong contrasts (e.g. "however", "but", "very", "likely") or topics of general interest (e.g. "security").

**Insights**

LIME offers a user-friendly and intuitive way to interpret model decisions. While it may not always capture the most precise token importance due to its approximate, local nature, it is extremely effective at communicating why a model made a certain prediction. Its strength lies in its visual and accessible explanations, which make it especially valuable in non-technical settings who need to understand and trust the model's decisions without requiring deep expertise in machine learning.

Notice the probabilities values: 1.00 for correct fake predictions and a very close difference between class probabilities when misclassified. This shows how confident our model is, being very close in decision on harder examples.

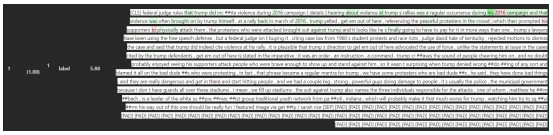
### 3.5.2 Integrated Gradients

Captum ("comprehension" in Latin) is an open source, extensible library for model interpretability built on PyTorch. We've used the "Primary Attribution" feature that Captum provides in order to try to see which part of the input text contributes the most to the model output - *real* or *fake* news. We used the **Integrated Gradients** algorithm provided by the library hoping to extract what we wanted: insight on what exactly is the fake part of the text. Integrated gradients represents the integral of gradients with respect to inputs along the path from a given baseline to input(Facebook, 2025).

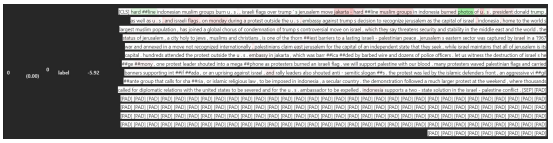
**Results Analysis**

We used the previously presented BERT model for predicting the probabilities, then using the Integrated Gradients feature provided by Captum, we tried to visualize the attribution score for each token derived from the text. A green colored token means it had a bigger contribution to the prediction, red - it impacted it negatively and no color means the model considered it to have a neutral contribution.

We chose two correctly identified news and 1 misclassified:

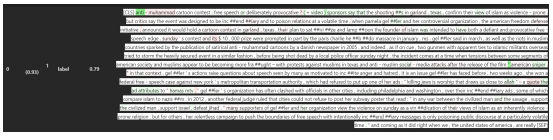


(a) Correctly classified real news



(b) Correctly classified fake news

(c) Misclassified real news



(d) Misclassified fake news

For real news correct prediction, the model seems to be mostly influenced by words such as

"campaign", "2016", "trump". For the fake news correct prediction, it's fake classification is influenced by "jakarta", "muslim", "groups", "president", "flags", strongly politically associated words, however it was positively influenced of the word "photos".

It is interesting to notice that for a baseline that is missing the **[CLS]** token (a special BERT token that contains the meaning of the whole text), the token attribution algorithm gives a lot of importance to that token and to other special tokens such as **[SEP]**, for example:

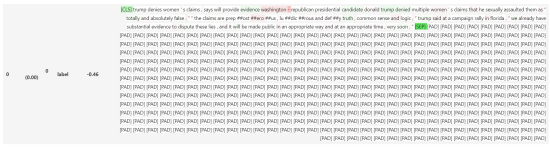


Figure 13: Correctly classified fake news with [CLS] token attribution

## 4 Future Work

Future Work may include:

1. **Sentiment analysis**- Incorporating sentiment analysis could provide insights into the predominant emotional tone of fake news articles, which may help distinguish them from legitimate reporting and reveal underlying persuasive strategies.

2. **Multilingual Support**- Addressing the current language limitations by extending the model to support multiple languages would enhance its applicability across diverse populations and reduce linguistic bias.

## 5 Conclusion

In this study, we explored multiple approaches for detecting fake news using natural language processing techniques. Our experiments ranged from traditional machine learning models such as SVM and Random Forest, utilizing Word2Vec embeddings, to state-of-the-art Transformer architectures like BERT and its variants. We also conducted exploratory data analysis using word clouds and bigrams to better understand linguistic patterns associated with misinformation.

While our models achieved promising results, we acknowledge the limitations inherent to this task, including dataset bias, linguistic ambiguity, and the evolving nature of deceptive content. Nevertheless, our work contributes to the growing body of research aimed at enhancing media literacy and developing tools to support the identification of disinformation.

## Limitations

Despite promising results, fake news detection using NLP faces several limitations. The definition of 'fake' is often subjective, with some content falling into gray areas such as satire or biased reporting. Models can struggle to generalize across different topics, writing styles, or time periods, especially as fake news strategies evolve. Additionally, understanding deception often requires external context, which purely text-based models may lack. Challenges like class imbalance, sarcasm, and limited explainability further complicate reliable and interpretable detection.

## Ethical Statement

The development of fake news detection systems, while aimed at promoting truth and media literacy, can also be misused. One potential unethical use of our research is the application of the system for censorship or political suppression, where it could be used to label dissenting or minority viewpoints as 'fake'. Additionally, automated systems risk overreliance, where users may accept model predictions as absolute truth without critical thinking.

## References


Pîrvulescu Daria-Maria Apostol Alin-Constantin, Ciuperceanu Vlad-Mihai. 2025. The github of the project. https://github.com/Apostol-Alin/NLP-fake-news-detection/tree/main. Last accessed: [2025-04-30].

Alberto Barrón-Cedeno, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9847–9848.

Ganesh Gopal Devarajan, Senthil Murugan Nagarajan, Sardar Irfanullah Amanullah, SA Sahaaya Arul Mary, and Ali Kashif Bashir. 2023. Ai-assisted deep nlp-based approach for prediction of fake news from social media users. *IEEE Transactions on Computational Social Systems*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the*

*North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Inc. Facebook. 2025. Captum official documentation. Last accessed: [2025-05-03].

Rohit Kumar Kaliyar, Anurag Goswami, and Pratik Narang. 2021. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia tools and applications*, 80(8):11765–11788.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you? explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Saurabh Shahane. 2021. Fake news classification dataset. https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification. Last accessed: [2025-04-29].

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3319–3328. PMLR.

Pawan Kumar Verma, Prateek Agrawal, Ivone Amorim, and Radu Prodan. 2021. Welfake: Word embedding over linguistic features for fake news detection. *IEEE Transactions on Computational Social Systems*, 8(4):881–893.