# Fake News Detection Using Large Language Models

Amr Morsy, Muhammad Sharafeldin, Nouha Jegham, Ranim Boussema

***Abstract*—Addressing the intricate challenge of fake news detection, traditionally reliant on the expertise of professional fact-checkers due to the inherent uncertainty in fact-checking processes. We compare different machine learning, deep learning and LLM models' performance on fake news detection task. This research proposes using large language models as feature extractors instead of using them end-to-end for this task. The evaluation is conducted on the well-established LIAR dataset, a prominent benchmark for fake news detection research. Our study presents detailed comparisons and ablation experiments on the LIAR. The proposed approach aims to enhance our understanding of dataset characteristics, contributing to refining and improving fake news detection methodologies.**

***Index Terms*—**

## I. Introduction

The growing popularity of social media platforms and the diminishing role of traditional news channels [1] present a significant challenge in combating the rampant problem of mis/disinformation in the modern day [2]. With thousands of data getting shared per day in several shapes —whether it be text, audio, or video forms — verifying and validating this data have become integral. Typically, fake news is defined either as false news, or a news article that has been purposely and verifiably falsified administratively [3][4]. Studies show that fake news is spreading faster on social media [5], and generalization of fake news is very hard to control [6].

The prevalence of misinformation poses significant risks. In 2016, an individual armed with an AR-15 rifle launched an attack on a pizzeria in Washington, D.C. This violent act was motivated by his belief in a fabricated narrative claiming that the establishment was a front for a child trafficking operation involving Hillary Clinton. Subsequently, law enforcement apprehended the assailant, who faced charges for discharging an assault weapon within the premises of the restaurant [7]. Furthermore, misinformation can also influence political outcomes; research indicates that the choices made by voters during the 2016 U.S. election were swayed by false information [8].

Due to the need for specialists, manual verification of misinformation is a lengthy task, and one of the factors leading to slow responses[2]. Studies show that laypersons find this difficult; when asked to identify false statements, human evaluators have only succeeded between 50% to 63% of the time [9]. In addition, another study showed that subjects rated a misleading news article as "somewhat" or "very" accurate 3 quarters of the time [10], and an additional study found that ˜80% of high school students have trouble distinguishing the truth of news articles [11].

One of the most challenging issues in today's world is the enhancement of an efficient automated model for the detection of fake news [2]. Utilizing the advances now available in artificial intelligence, we can better segregate fake news from reality using systems that detect it automatically; The increasing attention in this area is seen through the numbers of publications on this line of research in recent years, as noticed in a survey held by Farhangian et al. [2] as shown in figure 3.

This paper studies LIAR benchmark and it applies standard ML methods such as Random Forest, support vector machines, logistic regression. It also employs deep learning models such as Convolutional Neural Networks (CNN) and Bi-Directional Long Short-Term Memory (BiLSTM) networks for feature extraction. We also made use of large language models (LLMs like GPT-3). 5, LLaMA, and Gemini, and using Gecko large language models [12] as feature extractors of a neural network.

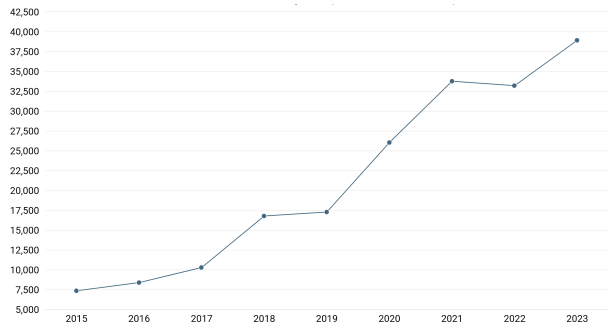


Figure 1. Number of publications related to fake news detection source: app.dimensions.ai

## II. Related Work

### A. Feature Extraction

Textual features are defined as features derived from the headline or text body from news articles. There are three types of textual features: Linguistic features, Low-rank textual features, and Neural textual features. Linguistic features examine the written text on various levels (e.g. words, and sentences). Lexical, syntactic, and semantic features are examples of linguistic features. Low-rank models apply tensor factorization to obtain a small-scale text representation from a high-dimensional and sparse input feature matrix [3]. Neural Networks with textual characters are represented by vectors. Some methods (e.g., Word2Vec, GloVe, and FastText) are context-independent [13]. Others provide contextual informa-

tion, such as Language-models like Chatgpt [14] or BERT [15]

Truică et al. [16] note that document embeddings (DOCEMBs) play a pivotal role in fake news detection, and that the selection of embedding strategy is, in many cases, more important to performance than the complexity of their classification model. The authors used embeddings like BART, ROBERTA, TF-IDF with simple neural architectures such as BiGRU/BiLSTM to achieve state-of-the-art performance; surpassing complex Deep Neural Networks and matching a more advanced model performance on large news corpora.

### B. Models

Large Language Models, Deep Learning, and Classical Machine Learning were the three categories into which the models were divided.

*1) Classical Machine Learning:* There is a rich variety of classical machine learning algorithms, including probabilistic models (such as Naive Bayes and Logistic Regression) as well as memory-based models (e.g., K-nearest neighbors and Support Vector Machines), and rule-based models (e.g., Decision Trees) [13]. In their study, Reddy et al. [17] applied multiple classical machine learning algorithms to the McIntire [18] and FakeNews [19] datasets to classify fake news. These datasets comprise political news articles ranging from left-leaning to right-leaning perspectives in relation to the 2016 US elections. They performed experiments using RF, KNN, Naive Bayes, LR, and SVM, achieving the highest accuracy of 82.5% using RF. The results are displayed in Table. I.

Table I
CLASSICAL MACHINE LEARNING ALGORITHM RESULTS ON THE MCINTIRE[18] AND FAKENEWS [19] DATASETS [17]

| Model Name | Accuracy |
|---|---|
| **Rndom Forest** | **82.5%** |
| Naive Bayes-Multinomial | 67% |
| Naive Bayes-Gaussian | 70% |
| SVM | 53% |
| Logistic Regression | 75.7% |

*2) Deep Learning:* Deep Learning is one of the most successful approaches to building news classification systems. There were several models proposed to solve the problem of fake news detection using Deep Learning techniques. Wang et al. [20] introduced the LIAR dataset and proposed a hybrid CNN-LSTM model to improve the classification of news text with context information. This model achieved an accuracy rate of 27.4% on the LIAR dataset, serving as a baseline performance.

In their research, Liu et al. [21] used BERT model [15] as the embedding layer but it uses a better transformer to extract the features instead of CNN or LSTM networks. They created a two-step mechanism where the first step puts inputs into a coarse binary classification and the model output from this task becomes an input to another model that refines these into finer classes. They achieved accuracy rates of 34.51% without contextual data and 40.58% using contextual data.

Xu et al. [22] proposed a fuzzy CNN-BiLSTM network configuration, wherein, state-of-the-art results were reported in multiclass classification on the LIAR dataset. Their architecture consisted of two parts: one that dealt with text features and another that dealt with numerical features, which were subsequently combined to produce predictions. On the LIAR dataset, this method reached a 46.5% accuracy.

*3) Large Language Models:* Pre-trained Language Models (PLMs) have gained more and more preference for its exceptional efficiency on various Natural Language Processing (NLP) tasks. When these models grew to include a significant number of parameters, they were eventually called Large Language Models (LLMs) [23]. Because these tasks inherently rely on language, LLMs are heavily involved in addressing such challenges like fake news detection, sentiment analysis, and bias detection, leveraging their ability to embed linguistic features in text representations.

Grover was introduced in [24], Grover is a Language Learning Model (LLM) trained with a Generative Adversarial Network (GAN) to produce and identify fake news. Grover, which was based on the 1.5 billion parameter GPT-2 model [25], showed that including meta-data along with text during the training led to better performance and also that performance improved with increased model size. Similarly, Cheung et al. (2023) [26] trained LLaMA with additional capabilities (e.g. integrate Google API for evidence retrieval) into FactLLaMA. FactLLaMA results were 29% without evidence retrieval and 30% with evidence retrieval, highlighting the impact of context in recognizing fake news.

Voyage-3-large [27], is a state-of-the-art general-purpose and multilingual embedding model that ranks first across eight evaluated domains spanning 100 datasets, including law, finance, and code. It ranks number one in the Massive Text Embeddings Benchmark (MTEB) for classification tasks.

Farhangian et al. (2023), [2], performed a comprehensive analysis in the domain of automatic fake news detection and proposed an improved taxonomy, which considers different feature types, perspectives of detection, representation techniques, and classification methods. Their empirical analysis of 15 feature extraction methods incorporated traditional count-based approaches as well as advanced word embeddings and transformer models. They also benchmarked 20 classification algorithms including Support Vector Machines (SVM), Bidirectional Long Short-Term Memory networks (BiLSTM), and ensemble learning techniques.

As shown in Table II, transformer-based models such as LLaMA, Falcon and BART significantly outperformed other model types when playing the role of feature extractor. We also found that the combination of LLaMA and SVM had an absolute highest F1-score of 26.6%. These transformer models outweighed end-to-end setups in accuracy and versatility while enabling a tabulated diversity and smooth accessibility to complementary classifier architectural ensembles.

Crucially, fusing five to six various feature extraction strategies with a fixed classifier (i.e., SVM) provided solid generalization and theoretically allowed for perfect accuracy

on different datasets. This not only proves the importance of incorporating a diversity of feature spaces while maintaining efficacy but also demonstrates how both old and new methods of processing text can improve systems aimed at preventing fake news from circulating.

Boissonneault et al. (2024) in [28] evaluated ChatGPT and Google Gemini using the LIAR dataset, achieving binary classification accuracies of 89% and 91% respectively, exceeding existing state-of-the-art metrics. Using a wide range of metrics, from precision and recall to F1 and AUC-ROC, their study drew attention to the practical usefulness of these models in verifying information in fact-checking systems, achieving solid benchmarks.

To show detailed performance evaluation on the LIAR dataset, literature review results for classification tasks regarding binary and multi-class are included in Table III and IV, respectively.

## III. Research Questions

Q1: Does the performance of LLM exceed that of conventional approaches in detecting fake news?

Q2: Does the implementation of a novel architecture that utilizes Transformers for feature extraction, in conjunction with a FuzzyLayer, enhance overall performance?

## IV. Datasets

Detecting fake news using machine learning has led to the development of multiple datasets due to the need to counter the proliferation of misinformation. An initial effort in this direction has been made by Vlachos and Riedel [30], which created a dataset of fact-checking data from two well-known websites that contain only 221 statements, but it wasn't enough to meaningfully develop machine learning models. To address these limitations, Wang [20] proposed the **LIAR** dataset, a comprehensive collection from PolitiFact.com, throughout 12,836 short stances, following a hand-defined attribute. This corpus contains statements from 2007 to 2016 across a variety of venues, including statements from political debates, television ads, social media platforms, and official press releases.

Each entry in the LIAR dataset is annotated with multiple labels, including a six-way classification of truthfulness: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*. Entries are also tagged by subject, speaker affiliation, context, and historical backdrop. Each classification is validated by a thorough analysis report and a respective source. Such multi-faceted labelling results in ample structured data that have proven useful to build predictive models capable of detecting complicated patterns of misinformation and the fine linguistic detail that denotes untruthfulness.

Another important dataset is the Fact Extraction and Verification **FEVER** dataset proposed by Thorne et al. [31]. This dataset includes a total of 185,445 claims that have been externally labelled as *SUPPORTED*, *REFUTED* or *NOTENOUGHINFO* label endowing an evidence-based description of the veracity of the claims. While FEVER does offer an extensive set of claims, it does so with a strong focus on binary classifications. On the other hand, the LIAR dataset uses multi-class labeling which should be suitable for ML/DL models with softmax layers that better fit the gradual nature of truth.

The **COVID-19** fake news dataset to address misinformation regarding the COVID-19 pandemic was developed by Shahi and Nandini [32]. This dataset contains 10700 entries labelled as either real or fake news. News, mostly from Twitter, each item was verified by credible outlets like PolitiFact.com and Snopes.com, the research on MERSVIRUS tracks misinformation from Twitter, a platform with shorter and faster-moving messages than platforms that have been studied for pandemic misinformation overall.

**FakeNewsNet** dataset was created by Shu et al. [19] to include a wider variety of online misinformation types. It merges news articles from PolitiFact and GossipCop with rich metadata like user engagement metrics and source reliability indicators. FakeNewsNet has been created with over 23,000 articles that are classified into fake and real news and is capable of allowing researchers to explore misinformation in various dimensions such as social context and propagation dynamics, especially on platforms like Twitter. The rich metadata of this dataset makes it comparatively well-placed in exploring how misinformation propagates and interacts with user behaviours online.

LIAR corpus has an average statement length of 17.9 tokens, which is very appropriate for detecting misinformation in the type of compact content in the news domain. On the other hand, the COVID-19 dataset is centred around pandemic-based concerns and does tackle issues of public health misinformation. FakeNewsNet also covers user interaction patterns, which characterize the spread and consumption patterns when fake news occurs on social media platforms. However, despite their extensive tagging and annotation, the output of models trained on them may not translate well to real-world environments, given the disparity between curated datasets and the conventional aspects of live media environments. These datasets effectively serve as the backbone for developing automated tools for fake news detection, which is critical for curbing the rapid spread of misinformation on digital platforms. Diverse in their architectures, from multi-class analysis `(LIAR)`[20] to evidence-based verification `(FEVER)` [31], misinformation recognition `(COVID-19)` [19], or social context `(FakeNewsNet)` [30][20], such collections offer a selection of methodologies to be refined towards leveraging machine-learning to retain the integrity of information.

## V. Pre-processing

### A. Pre-processing

Various pre-processing steps were performed on the LIAR dataset. This process helped to ensure that the data quality was high, the model's performance was better, and the text representation was consistent.

*1) Preprocessing of Textual Features:* The preprocessing setup for the LIAR textual features (as specified by Columns 2, 3, 4, 5, 6, 7, 8 and 14 in Table VI) applied was:

Table II
AVERAGE F1-SCORE AND STANDARD DEVIATION OBTAINED ON THE LIAR DATASET. EACH ROW REPRESENTS A FEATURE REPRESENTATION TECHNIQUE, WHILE EACH COLUMN REPRESENTS A CLASSIFICATION MODEL. THE LAST COLUMN SHOWCASES THE PERFORMANCE OF END-TO-END TRANSFORMER MODELS. THE BEST RESULTS FOR EACH FEATURE REPRESENTATION ARE HIGHLIGHTED IN BOLD. THE CELLS MARKED WITH AN (*) INDICATE THAT THE CLASSIFICATION MODELS AND FEATURE REPRESENTATION TECHNIQUES IN THE CORRESPONDING ROWS AND COLUMNS ARE INCOMPATIBLE. (ADAPTED FROM FARHANGIAN ET AL., 2024[2].)

| Features | **SVM** | **LR** | **KNN** | **NB** | **RF** | **AdaBoost** | **XGBoost** | **MLP** | **CNN** | **BiLSTM** | **End-To-End** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **TF** | **25.6%** | 24.5% | 21.9% | 24.2% | 24.6% | 22.3% | 28.8% | 22.6% | * | * | * |
| **TF-IDF** | **24.6%** | 24.2% | 22.8% | 23.9% | 23.6% | 21.8% | 22.0% | 22.2% | * | * | * |
| **Word2Vec** | **24.8%** | **24.8%** | 22.3% | 21.9% | 25.0% | 24.4% | 23.5% | 22.8% | 22.5% | 22.4% | * |
| **GloVe** | **24.2%** | 24.0% | 21.2% | 22.5% | **24.2%** | 23.7% | 22.8% | 23.5% | 21.8% | 22.9% | * |
| **FastText** | **24.5%** | 24.2% | 22.2% | 21.8% | 24.3% | **24.5%** | 23.9% | 23.3% | 22.6% | 24.2% | * |
| **ELMO** | **26.4%** | 25.8% | 23.2% | 21.8% | 25.6% | 25.3% | 24.9% | 23.0% | 22.1% | 23.1% | 21.0% |
| **BERT** | **25.6%** | 25.3% | 22.0% | 22.5% | 23.7% | 23.7% | 21.7% | 20.9% | 19.0% | 21.0% | 21.0% |
| **DistilBERT** | **25.2%** | 22.4% | 21.0% | 22.3% | 24.8% | 24.0% | 22.2% | 21.6% | 19.0% | 21.0% | 22.0% |
| **ALBERT** | **24.7%** | 22.8% | 20.5% | 22.1% | 23.0% | 23.4% | 22.4% | 21.8% | 23.0% | 24.0% | 18.0% |
| **BART** | **25.5%** | 23.6% | 21.7% | 22.6% | 24.1% | 24.9% | 22.0% | 20.9% | 21.0% | 23.0% | 20.0% |
| **RoBERTa** | 23.8% | 23.8% | 20.3% | 22.6% | 23.2% | 24.1% | 22.1% | 21.8% | **25.0%** | **25.0%** | 19.0% |
| **ElECTRA** | **24.1%** | 22.2% | 19.7% | 21.8% | 23.1% | 22.9% | 21.5% | 20.6% | 22.0% | 23.0% | 19.0% |
| **XLNET** | 21.7% | 22.1% | 19.6% | 21.8% | 22.8% | **23.7%** | 21.6% | 20.5% | 21.0% | 22.0% | 21.0% |
| **LLaMA** | **26.6%** | 24.6% | 22.6% | 22.0% | 25.2% | 24.4% | 24.2% | 24.4% | 25.2% | 25.8% | 26.1% |
| **Falcon** | **25.8%** | 25.1% | 22.7% | 22.0% | 24.1% | 22.3% | **26.1%** | 24.2% | 25.0% | 25.6% | **25.8%** |

Table III
MODELS ACCURACY ON THE LIAR DATASET IN MULTICLASS CALSSIFICATION

| **Model Name** | **Accuracy** |
|---|---|
| SVM (Wang et al., 2017) [20] | 25.5% |
| LR (Wang et al., 2017) [20] | 24.7% |
| CNN-BiLSTM (Wang et al., 2017) [20] | 27.4% |
| Two-Stage BERT (Liu et al., 2019) [21] | 34.5% |
| Two-Stage BERT with context (Liu et al., 2019) [21] | 40.5% |
| **Fuzzy CNN-BiLSTM (Xu et al., 2024) [22]** | **46.5%** |
| FactLLaMA without context (Cheung et al., 2023)[26] | 29.9% |
| FactLLaMA with context (Cheung et al., 2023)[26] | 30.4% |

Table IV
MODELS ACCURACY ON THE LIAR DATASET IN BINARY CALSSIFICATION

| **Model Name** | **Accuracy** |
|---|---|
| Bert Base + Feedforword NN (Upadhayay et al., 2020) [29] | 69% |
| Bert Base + CNN (Upadhayay et al., 2020) [29] | 70% |
| ChatGPT (Boissonneault et al., 2024) [28] | 89% |
| **Goolge Gemini (Boissonneault et al., 2024) [28]** | **91%** |

- **Removal of Quotations:** Removed all quotation symbols from the textual content.
- **Punctuation Removal:** All the characters present in the string of the following form were removed to reduce noise:

  `!"#$%&'()*+,-./:<>=?@[\]^_`{|}~`

- **Stop Word Removal:** Removal of common high-frequency stop words like *"the," "a," "is,"* and other words from a predefined list. This technique minimizes noise and helps the model learn meaningful content.
- **Normalization:** All uppercase letters were transformed to lowercase to maintain consistency and avoid duplicate representations of words based on case.
- **Sentence Boundary Marking:** Each sentence was prepended with a *Start of Sentence (SOS)* token and appended with an *End of Sentence (EOS)* token to mark sentence boundaries.
- **Dealing with Missing Values:** Missing values were handled in two ways:
  - **Drop Null Values:** Null values in a data entry led to it being dropped when required.
  - **Textual Null Values Replacement:** Textual features with null values were replaced with an `<UNK>` token to preserve the data structure.

*2) Tokenization & Sentence Padding:* The text was first cleaned and later tokenized into words, which served as input for vocabulary creation and numerical representation. To efficiently process inputs in batches, each sentence was padded with the `<PAD>` token to match the length of the longest sequence in the batch.

*3) Text Normalization and Noise Filtering:* Other text normalization techniques were then executed to improve the dataset further:

- **Lowercasing:** Brought uniformity to the text by changing all characters to lowercase, minimizing vocabulary size, and enhancing model efficiency.
- **Removal of Stop Words:** This involved the removal of words whose contribution to the document's semantic meaning was minimal and irrelevant, thus allowing the filtering of only informative content.
- **Punctuation Removal:** All non-essential punctuation marks and syntactic elements were removed as they can serve as distractions during training.

*4) Dealing with Missing Data:* There were missing values in the LIAR dataset (as shown in Table V). Null values were dealt with using the following techniques:

- Textual features containing empty fields were replaced with an `<UNK>` token to ensure data consistency.
- Rows with "null" in the Historical Fact-Checking Count field were dropped from the dataset with little overall

impact on data distribution (only 2 rows were dropped; results not shown). The distribution of the labels in the LIAR dataset labels distribution is shown in Figure 2.

*5) LLM Considerations:* Another advantage of LLMs compared to more traditional word embedding models is the granularity at which text is handled, LLMs operate at the sentence level, whereas traditional models break text into words. Therefore, in the case of using LLMs, further text preprocessing was not performed. Some prompt examples provided to LLMs for their training are shown in Subsection VII-C3, where raw text data was used instead.

Table VI
DATASET COLUMN DESCRIPTIONS WITH EXAMPLE

| Feature | Description | Example |
|---|---|---|
| f1 | Unique ID of the statement (`[ID].json`) | `324.json` |
| f2 | Label of the statement (e.g., true, false, half-true) | mostly-true |
| f3 | Text of the statement | Hillary Clinton agrees with John McCain "by voting to give George Bush the benefit of the doubt on Iran." |
| f4 | Subject(s) of the statement | foreign-policy |
| f5 | Speaker of the statement | barack-obama |
| f6 | Speaker's job title | President |
| f7 | State information related to the statement | Illinois |
| f8 | Speaker's party affiliation | democrat |
| f9–f13 | Historical fact-checking counts:<br>• Barely true (9),<br>• False (10),<br>• Half-true (11),<br>• Mostly true (12),<br>• Pants-on-fire (13). | • 70 (Barely true),<br>• 71 (False),<br>• 160 (Half-true),<br>• 163 (Mostly true),<br>• 9 (Pants-on-fire). |
| f14 | Context (venue or location of the statement) | Denver |

## B. Data Loading

LIAR: A Benchmark Dataset for Fake News Detection dataset is divided into three subsets: training, validation, and test sets. A vocabulary was created for each subset, mapping each unique token derived from preprocessing to a unique identifier, to ease model training and evaluation. This ID is subsequently employed to represent the token in the embedding layer.

For both training and validation, the data is loaded into batches as needed to ensure computational efficiency. Because deep learning models expect a sequence of the same length as input, a custom function was built to pad each sentence with a padding token `<PAD>` so that all the sentences in a batch have the same length as the longest sentence.

Furthermore, during the inference process of our model, some words may be OOV (out of vocabulary) so we introduce a special unknown token `<UNK>`. It acts as a stand-in for any term not found in the vocabulary built from our dataset, which adds resilience for applying the model to fresh text input.

The data loading pipeline is explained, so the input sequences were processed efficiently, allowing for seamless training and inference of different models used for fake news detection.

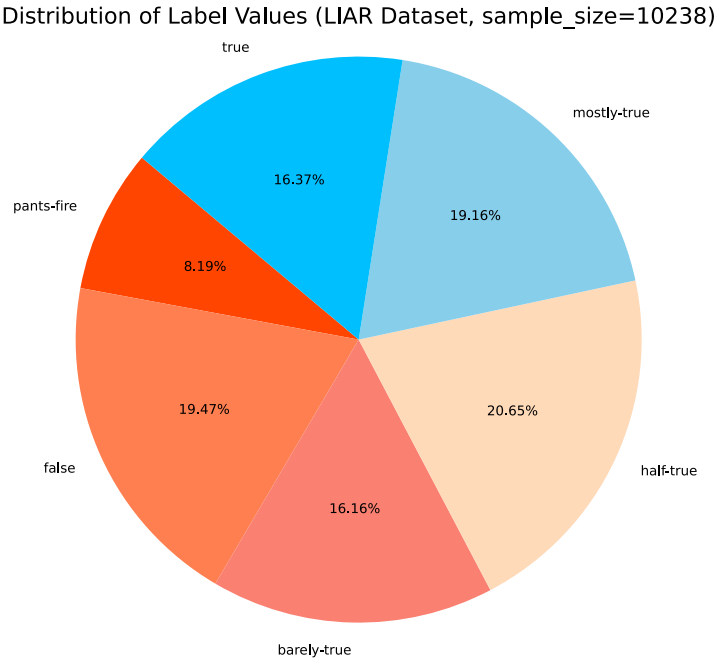


Figure 2. Distribution of label values in the LIAR dataset. Each class has a significant proportion of samples. However, there is a slight imbalance where the pants-on-fire category has the lowest proportion (8.19%) compared to the others.

From the pie chart, we observe that the dataset is relatively balanced, with each class having a significant proportion of samples. However, there is a slight imbalance where the *pants on fire* category has the lowest proportion (8.19%) compared to the others. The most frequent category is *half true* with 20.65%. These variations may impact model performance.

# VI. PROPOSED MODELS

In this research, we tried three different types of models — machine learning models, deep learning models, and large language models. We evaluated their performance given various features and with respect to diverse experiment settings.

## A. Machine Learning Models

This research implements a few classical machine learning models to have a comparison base for the performance of fake news detection. These ground truth labels in the LIAR dataset with preprocessed textual features are used for these models.

*1) Support Vector Machine:* Support Vector Machines are a class of supervised learning models and are popular for text classification tasks because of their efficiency in high-dimensional feature spaces. The study used a linear kernel

Table V
NUMBER OF NULL VALUES IN TRAIN, VALIDATION, TEST SPLIT.

| **Column Name** | **# Of nulls in Training** | # Of nulls in Val | # Of nulls in Testing |
|---|---|---|---|
| id | 0 | 0 | 0 |
| label | 0 | 0 | 0 |
| statement | 0 | 0 | 0 |
| subject | 2 | 0 | 0 |
| speaker | 2 | 0 | 0 |
| job_title | 2898 | 345 | 325 |
| state_info | 2210 | 279 | 262 |
| party_affiliation | 2 | 0 | 0 |
| barely_true_counts | 2 | 0 | 0 |
| false_counts | 2 | 0 | 0 |
| half_true_counts | 2 | 0 | 0 |
| mostly_true_counts | 2 | 0 | 0 |
| pants_on_fire_counts | 2 | 0 | 0 |
| context_venue_location | 2 | 12 | 17 |

SVM to classify statements into six truthfulness categories based on the LIAR dataset.

*2) Random Forest:* Random Forest is an ensemble machine learning algorithm that builds a multitude of decision trees from the training set and outputs the mean prediction of the individual trees. It builds tons of decision trees and aggregates the output from all the trees.

### *B. Deep Learning Models*

Our implemented neural network models displayed the same network architecture composed of three main components, and further details on the components are in the sub-sections that follow. Associated embedding component VI-B2, feature extraction component VI-B3, and classification component VI-B4. The over-all architecture is shown in Figure 3. We built various models for each with key components to analyze how the influence of each technique affected performance.

*1) Feature Categories:* Our deep learning models are based on the architecture of Xu et al. [22] which yielded the state-of-the-art performance. We have divided the input into three categories. The first one being the news statement, which is basically the written portion of the news article. It is used as the main source of data for false news detection. The second category is the context, which is the textual context of the statement, contains components like the identity and political party of the author specified by the required keys *f4-f8* and *f14* in Table VI. The third kind is the numbers providing context to the statement, describing the speaker's past behavior of issuing phony statements.

*2) Embedding Layer:* The embedding component transforms input text into vectors. These vectors are mapped into a vector space with semantically similar words physically mapped closer to each other inside that vector space. This transformation addresses the limitations of sparse one-hot encoding by reducing dimensionality. Transforming text into numerical vectors allows the rest of the network to do mathematical computations on the input text. For this research we used a linear embedding layer, Word2vec pretrained embedding layer, Voyage LLM, and Gecko text-embedding-04 embedding.

*3) Feature Extraction Module:* Feature extraction plays a crucial role in various natural language processing tasks. This study investigates the efficacy of three prominent deep learning architectures – Bidirectional Long Short-Term Memory (BiLSTM), Convolutional Neural Networks (CNN), and Large Language Models as feature extraction layers within a unified framework. BiLSTM excels at capturing sequential dependencies, CNNs effectively identify local patterns, and LLMs as feature extractors.

*4) Classification Layer:* The classification layer maps the outputs of the feature extraction layer to a probability distribution for the 6 labels. We use the Softmax function to map the outputs to a distribution such that every label has a value between 0 and 1, and all values sum up to one.

### *C. Large Language Models*

Large Language Models (LLMs) are advanced NLP models that use transformer architectures with self-attention mechanisms to process text efficiently. Unlike traditional RNNs, transformers handle entire sequences in parallel, enabling the training of large-scale models with billions of parameters. Pre-trained on massive datasets, LLMs excel in tasks such as text classification, content generation, and question answering. This study evaluates three LLMs; Gemini-1.5-Flash, GPT-3.5 Turbo, and LLaMA-3.

## VII. IMPLEMENTATION

In this section, we elaborate on the specific configuration of the models employed in the experiments. As well as experimental settings such as hyper-parameters, optimizers used, loss functions, and prompts for large language models.

### *A. Machine Learning Models*

- **SVM:** TF-IDF embedding was done on textual data (i.e., *News Statement* and *Text Context*). We tested three feature configurations: all numeric and text features combined, just the text features, and only the *Statement* column. Hyperparameters like *C*, *kernel*, and *gamma* were optimized using GridSearchCV, with five-fold cross-validation for optimal performance.

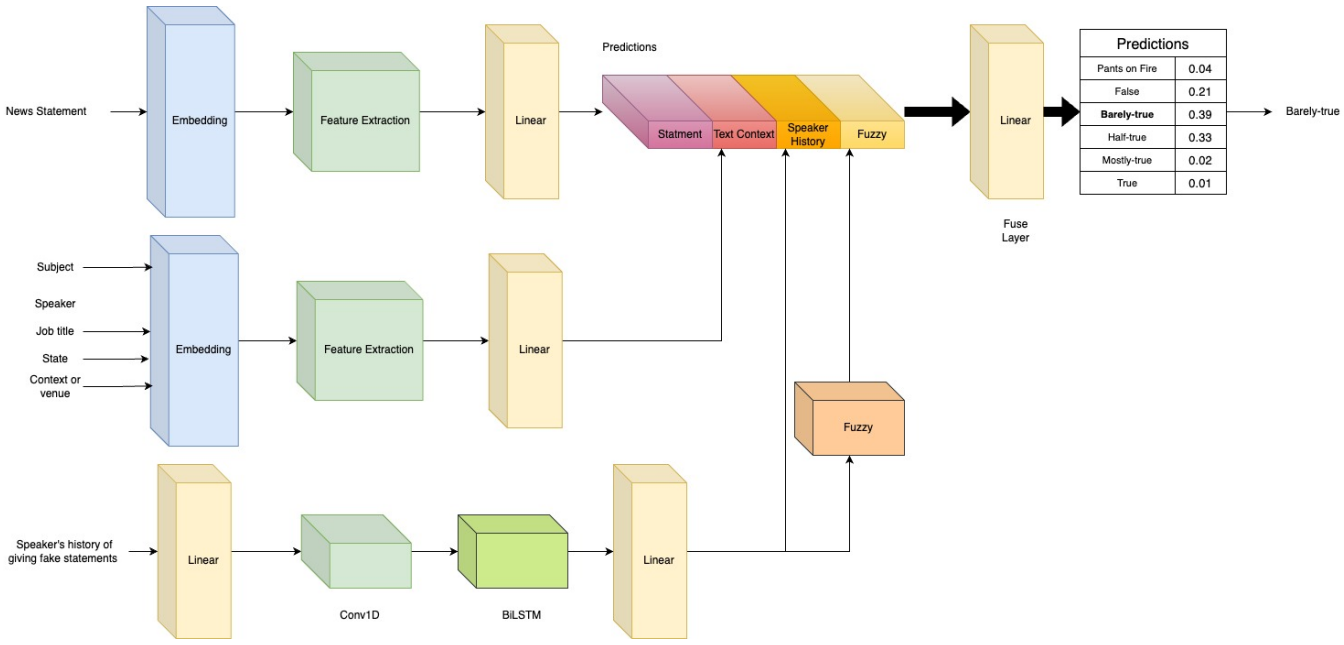


Figure 3. Deep learning model architecture

- **Statement Only:** In this configuration, only the *Statement* column, embedded through TF-IDF, is used; numeric and contextual features are excluded. The model cannot, however, rely on actual linguistic cues in the statements. Hyperparameters used were (*C*=0.1, *kernel*=rbf, *gamma*=auto).
- **Context Only:** In this configuration, we exclude numeric features and use only the text features from the *Statement* and *Context* columns. With only the TF-IDF embedded text data as input, it pays all attention to linguistic and contextual parts. The hyperparameters used were (*C*=10, *kernel*=linear, *gamma*=scale).
- **All Context and Numeric Features:** This model integrates TF-IDF embedded text features of *Statement* and *Context* with numeric features such as *Barely_True_Count* and *False_Count*. The SVM model is semi-supervised and learns from both language and statistics. Hyperparameters used were first tested on (*C*=1, *kernel*=rbf, *gamma*=scale).

- **Random Forest Model:** For the *Statement* and *Context* columns, the text data was embedded using TF-IDF. Thus, we tested three feature configurations: combining text and numeric features; text features only; and the *Statement* column only. Hyperparameters such as *n_estimators*, *max_depth*, and *min_samples_split* were optimized using GridSearchCV with five-fold cross-validation.
  - **Statement Only:** This setup utilizes only the *Statement* column in embedded TF-IDF format, without numeric and contextual features. The model has no way of knowing whether the statements are true, and it bases its analysis entirely on linguistic patterns. Hyperparameters used: (*max_depth*=30, *min_samples_leaf*=1, *min_samples_split*=2, *n_estimators*=100).
  - **Context Only:** This configuration excludes numeric features and only uses text features present in the *Statement* and *Context* columns. It relies largely on linguistic and contextual semantic embedding using TF-IDF embedded text data. The hyperparameters were (*max_depth*=30, *min_samples_leaf*=2, *min_samples_split*=2, *n_estimators*=200).
  - **All Context and Numeric Features:** The features in this configuration are TF-IDF embedded text features of *Statement* and *Context* combined with numeric features such as *Barely_True_Count* and *False_Count*. Using both linguistic and statistical patterns, the Random Forest model trains on the dataset to provide firm classifications and strong predictions. The hyperparameters used were (*max_depth*=40, *min_samples_leaf*=2, *min_samples_split*=5, *n_estimators*=100).

### *B. Deep Learning Models*

The implementation of all models is carried out using PyTorch. Executed on NVIDIA GeForce RTX 3090 GPU. The output sequence length of each module is set to 6, the dropout rate is configured at 0.5. Cross-entropy loss was used as a loss function. Adam was used as the optimizer with a learning rate of 0.001 and weight decay of 0.0001. All models were trained for 300 epochs with a batch size of 32. To process the numerical context, A hybrid

CNNBiLSTM architecture inspired by Xu et al. [22] was used. The input features are concatenated and then fed to a linear layer with an input dimension of five and an output dimension of 16. The output is then fed to a CNN layer with a kernel size of one and an output channel size of 32. The output of the CNN layer is then fed to a BiLSTM layer with a hidden size of 64. The output of the BiLSTM layer is then processed by a linear layer with an output of 6 creating the output sequence for the labels. The output sequence is then fed to the fuzzy layer and the fuse layer. The output of the fuzzy layer is also fed to the fuse layer.

All models used the same classification layer for each input category, composed of a linear layer with dimensions [Feature Dimensions, 6]. Models that used multiple input categories had a linear layer combining the different predictions from each module into a single prediction.

*1) BiLSTM Models:* For feature extraction, we used a BiLSTM Layer with a hidden dimension of 128 with a dropout of 0.2. A linear layer with an input dimension of 256 mapped the output of the BiLSTM layer to a sequence of 6 numbers, Softmax was used to map the output of the linear layer to a probability distribution for each label. This model had two variations for the news statements input category and the text context input category. A fuzzy variation was also implemented by adding the fuzzy module to process the numerical context along side the text modules.

- **BaseLine Model:** This model was trained only on the new statements as an input. The baseline model with Pytorch embedding layer that trains a tensor of size (Vocabulary size, Embedding Vector Dimension) that works as a lookup table of embeddings for each word. The vocabulary size was the size of unique words in the liar dataset after pre-processing which is 23053. An embedding dimension of 300 was chosen for this model. The embeddings in the lookup table are initialized randomly and learned through the training loop.
- **Word2vec Model:** This model used Word2Vec [33] pre-trained embedding layer, that was trained on the Google News dataset (about 100 billion words), with an embedding dimension of 300 to warm start the Pytorch embedding layer.

*2) TextCNN Models:* Convolutional Neural Networks (CNNs), originally designed for image processing, have emerged as a powerful alternative for text classification. By leveraging convolutional filters to extract contextual patterns, CNNs offer an effective means of extracting features from a text sequence. These models used an architecture inspired by Xu et al...[22] implemented three parallel CNNs with kernel sizes [3,4,5] and an output channel size of 128. A linear layer with Softmax was used for classification with an output dimension of 6.

- **BaseLine Model:** This model was trained only on the new statements as input. The baseline model with Pytorch embedding layer trains a tensor of size (Vocabulary size, Embedding Vector Dimension) that works as a lookup table of embeddings for each word. The vocabulary size was the size of unique words in the liar dataset after pre-processing which is 23053. An embedding dimension of 300 was chosen for this model. The embeddings in the lookup table are initialized randomly and learned through the training loop.
- **Word2vec Model:** This model used Word2Vec [33] pre-trained embedding layer, that was trained on the Google News dataset (about 100 billion words), with an embedding dimension of 300 to warm start the Pytorch embedding layer.

*3) LLMS as feature extraction:* For this category of models, we experimented with using two LLMs for feature extraction for textual input: Gecko[12] and Voyage[27], producing an output dimension of 768 and 1024 respectively, for the classification layer.

### *C. Large Language Models*

In this sub-section we discuss using prompts to test the fake news detection capabilities of generative large language models.

*1) Overview of Prompts and Usage:* A prompt is the input or instruction provided to a Large Language Model (LLM) to guide its behavior and generate desired outputs. The quality of a prompt directly influences the accuracy and relevance of the model's output. Well-crafted prompts can improve performance across tasks like classification, reasoning, and content generation. Prompts typically include a task description, context, and optional examples to illustrate the expected response format. They can be used in various configurations, such as:

*2) Learning Configurations:*

- **Zero-Shot Learning:** In zero-shot learning, the model performs tasks without any prior task-specific training or examples. It relies solely on its pre-trained knowledge and contextual understanding of the input prompt. This configuration evaluates the model's ability to generalize to new tasks based only on instructions.
- **One-Shot Learning:** In one-shot learning, the model is provided with a single example of the task as part of the input. This example serves as a guide for the model, enabling it to understand the task's requirements and expected output format.
- **Two-Shot Learning:** Two-shot learning extends one-shot learning by including two examples of the task. The additional example enhances the model's understanding of task-specific nuances, often leading to improved performance compared to zero-shot and one-shot configurations.

*3) Prompt Examples:*

**Zero-Shot Prompt**

**Task:** You are tasked with analyzing the following article. Perform the following steps:

1) Evaluate the article's truthfulness based on the categories: `True`, `Barely True`, `False`, `Half-True`, `Mostly True`, `Pants-on-Fire`.
2) Justify your evaluation by identifying specific facts or claims from the article that influenced your decision.

**Message:** [The Statement]
**Output Format:**

**Truthfulness:** [One of the categories]
**Reasoning:** [Detailed justification based on the article's content]

**One-Shot Prompt**

**Task:** You are tasked with analyzing the following article. Perform the following steps:

1) Evaluate the article's truthfulness based on the following categories:
   - True
   - Barely True
   - False
   - Half-True
   - Mostly True
   - Pants-on-Fire
2) Justify your evaluation by identifying specific facts or claims from the article that influenced your decision.

**Example:**
**Article:** "Says the Annies List political group supports third-trimester abortions on demand."
**Truthfulness:** False
**Reasoning:** The statement misrepresents the stance of Annies List, which does not explicitly advocate for third-trimester abortions on demand.

**Now, analyze the following statement:**
**Message:** [The Statement]
**Output Format:**

**Truthfulness:** [One of the categories]
**Reasoning:** [Detailed justification based on the article's content]

**Two-Shot Prompt**

**Task:** You are tasked with analyzing the following article. Perform the following steps:

1) Evaluate the article's truthfulness based on the following categories:
   - True
   - Barely True
   - False
   - Half-True
   - Mostly True
   - Pants-on-Fire
2) Justify your evaluation by identifying specific facts or claims from the article that influenced your decision.

**Examples:**
**Example 1:**
**Article:** "Says the Annies List political group supports third-trimester abortions on demand."
**Truthfulness:** False
**Reasoning:** The statement misrepresents the stance of Annies List, which does not explicitly advocate for third-trimester abortions on demand.

**Example 2:**
**Article:** "Hillary Clinton agrees with John McCain 'by voting to give George Bush the benefit of the doubt on Iran.'"
**Truthfulness:** Half-true
**Reasoning:** Clinton did vote on a measure related to Iran, but the context of the agreement with McCain is more nuanced and does not entirely align with the statement.

**Now, analyze the following statement:**
**Message:** [The Statement]
**Output Format:**

**Truthfulness:** [One of the categories]
**Reasoning:** [Detailed justification based on the article's content]

## VIII. Results and Analysis

In this section, we present the experimental results of three categories of models, Classical Machine learning, Deep learning and LLMs. We tested the performance of 28 models. experiments to test the effectiveness of different features, embeddings, and architectures in detecting fake news shown in Table VII. Initially, in the following sections we discuss the impact of training the models on different features. LLMs as a feature extraction demonstrated superior performance over other architectures, achieving noteworthy average accuracy of 44%.

Table VII
RESULTS OF ALL EXPERIMENTS, THE HIGHEST PERFORMING MODEL IN EACH FEATURE CATEGORY IS HIGHLIGHTED IN BOLD

| Model | Features | Embedding | # Shots | Mean Accuracy | STD |
|---|---|---|---|---|---|
| SVM | News Statement | Tf-Idf | - | 22.33 | 0.23 |
| Random Forrest | News Statement | Tf-Idf | - | 23.44% | 0.42 |
| BiLSTM | News Statement | Linear | - | 20.54% | 0.26 |
| BiLSTM | News Statement | Word2Vec | - | 21.88% | 2.76 |
| TextCNN | News Statement | Linear | - | 21.14% | 0.79 |
| TextCNN | News Statement | Word2Vec | - | 25.24% | 0.46 |
| **Gecko** | **News Statement** | **Gecko** | **-** | **30.42%** | **0.34** |
| Voyage | News Statement | Voyage | - | 28.16% | 0.40 |
| LLama | News Statement | - | Zero | 20.50% | - |
| GPT | News Statement | - | Zero | 26.70% | - |
| Gemini | News Statement | - | Zero | 20.99% | - |
| LLama | News Statement | - | One | 19.90% | - |
| GPT | News Statement | - | One | 25.30% | - |
| Gemini | News Statement | - | One | 20.84% | - |
| LLama | News Statement | - | Two | 13.60% | - |
| GPT | News Statement | - | Two | 24.30% | - |
| Gemini | News Statement | - | Two | 21.23% | - |
| SVM | New Statement + Text Context | Tf-Idf | - | 25.97% | 0.12 |
| Random Forrest | New Statement + Text Context | Tf-Idf | - | 25.40% | 0.92 |
| BiLSTM | New Statement + Text Context | Linear | - | 19.91% | 1.09 |
| TextCNN | New Statement + Text Context | Linear | - | 20.60% | 0.78 |
| Gecko | New Statement + Text Context | Gecko single embedding for both Features | - | 26.16% | 1.43 |
| **Gecko** | **New Statement + Text Context** | **Gecko separate embedding for each feature** | **-** | **27.52%** | **1.02** |
| SVM | New Statement + Text Context + Speaker History | Tf-Idf | - | 35.70% | 0.45 |
| Random Forrest | New Statement + Text Context + Speaker History | Tf-Idf | - | 37.60% | 0.24 |
| BiLSTM + Fuzzy CNNBiLSTM | New Statement + Text Context + Speaker History | Linear | - | 37.34% | 2.07 |
| TextCNN + Fuzzy CNNBiLSTM | New Statement + Text Context + Speaker History | Linear | - | 40.69% | 1.69 |
| Gecko + Fuzzy Linear | New Statement + Text Context + Speaker History | Gecko separate embedding for both Features | - | 42.84 | 0.65 |
| **Gecko +Fuzzy** | **New Statement + Text Context + Speaker History** | **Gecko separate embedding for each feature** | **-** | **44%** | **1.19** |

### A. Results on news statements only

This section discusses the results of all models run on the news statement feature shown in Table VIII. The best-performing model was the neural network model with Gecko LLM as a feature extractor, with a recorded accuracy of 30.42% and a standard deviation of 0.34. The neural network models with LLMs as feature extractors outperformed all other architectures with 30.42% average accuracy for gecko and 28.16% average accuracy for Voyage.

*1) Machine Learning models' results:* The Machine Learning models performed very similarly to each other, with SVM and Random forest having a mean accuracy of 23.34 and 23.44 respectively, and a standard deviation of 0.23 and 0.42 respectively. They out-performed the neural network models with no pertaining, and the BiLSTM model with Word2Vec pre-trained embedding layer.

*2) Deep Learning Models Results:* We experimented using three embedding techniques; linear layer which is the simplest of them, Word2Vec pretrained linear layer, and LLMs, namely Gecko's text-embedding-04 version and Voyage text-embedding-large. Three feature extraction modules were used, CNN, BiLSTM, and LLMs. All models had the same classification layer described in subsection VI-B. Pre-trained models outperformed the

models with no pre-training with an increased performance of 1.30% for BilSTM and 4.10% for TextCNN. The TextCNN models outperformed the BiLSTM models with and without pre-trained embedding. The Models using LLMs as feature extraction modules outperformed all other models with Gecko being the highest performing model with 30.42% accuracy and Voyage being the second highest with an average accuracy of 28.16%.

*3) LLMs Results:* Three generative LLMs were used in this experiment, GPT 3.5 Turbo, LLaMA-3, and Gemini flash 1.5. Three configurations for prompts were used; zero-shot, one-shot, and two-shot described in details in VII-C2. LLaMA and Gemini had comparable results with 20.5% and 20.99% accuracy for zero-shot learning, and 20.80% and 19.90% for One-Shot. All LLMs performed worse under a one-shot configuration than a zer-shot configuration. GPT was the highest-performing LLM with 26.70% average accuracy under a zero-shot learning configuration. GPT performed higher than all other models in this research except for deep learning models with LLMs as a feature extraction layer.

### *B. Results on news statement and text context as features*

This section discusses the results of models trained on statements and text-context features shown in Table VIII-B2. The average accuracy of machine learning models was improved by adding additional features. The performance of all the Deep learning models dropped significantly. Gecko was still the highest-performing model with an average accuracy of 27.52%.

*1) Machine learning models' results:* Both of SVM and RF models' performance improved by introducing the new textual context of the news statements. The performance increased by 3.6% and 2% respectively. This performance increase helped ML models outperform the deep learning models without the LLM as a feature extraction module and all the LLMs except for GPT.

*2) Deep Learning models' results:* All of the deep learning model's performance dropped after introducing the textual context features. Gecko still outperforms all other models with a mean accuracy of 27.52%. We also experimented with embedding both the textual context and news statements together using the same embedding with Gecko. This had a worse performance by 1.36%.

### *C. Results on all features*

This section discusses adding the speaker's history of giving fake news statements to the models' input features in training. All of the models' performance improved significantly. The performance increase ranged from 9.73% to 20%. Random forest and Fuzzy BiLSTM models had very similar results with 37.60% and 37.34% respectively. The best-performing model of all experiments was the Gecko model with a Fuzzy-CNNBiLSTM module for the numerical features. The results of all models are shown in Table X

*1) Machine learning models' results:* SVM model was the lowest-performing model with a mean accuracy of 35.70%, a 9.73% performance increase over using only the news statement and textual-context features. Random Forrest had an average accuracy of 37.60% increasing the performance from only using the news statement and textual-context features by a 12.20%.

*2) Deep learning models results:* All of the deep learning model's performance increased significantly after introducing the textual context features. We experimented with three variants of the Gecko model. The first model used a single embedding layer for all the features, including the speakers' history, this model achieved the worst performance out of the the three variants with 40.69% average accuracy. This is due to the speaker's history not being directly processed by the network. The other two variants had the same embedding for the news statements and textual-context. The Gecko Fuzzy Linear model had a single linear layer for processing the speakers' history and achieved a mean accuracy of 42.84%. The second model used the same CNNBiLSTM module as the other models, and it achieved the highest average accuracy of all models with an average accuracy of 44%.

## IX. Conclusion

This research introduces a new architecture utilizing LLMs as a feature extractors instead of an end-to-end solution to fake news detection task. Our approach distinguishes itself from conventional methods by integrating a classification algorithm rooted in fuzzy set theory, inherently equipped to handle uncertainties and imprecisions inherent in natural language contexts. Experimental results on the LIAR dataset showcase the superior performance of our approach compared to traditional machine learning, deep learning, and LLM approaches.

Our comprehensive analysis and experimentation demonstrated the performance of our novel approach on LIAR dataset and showcased the various effects of using different features. Our findings underscore the need for a balanced approach that leverages both statement and contextual information to enhance the robustness and interpretability of fake news detection models. This architecture opens doors to innovative research avenues, encouraging further exploration into using LLMs as feature extractors with different components to tackle the nuanced challenges of distinguishing fact from fiction. As the landscape of LLMs continues to evolve, newer models will open opportunities for researchers seeking to advance the frontiers of fake news detection.

## X. Future Work

Guided by the findings and insights from our current research, we envisage the following prospective avenues for future work:

- **Fine Tuning:** All of the Pre-trained LLMs used in this research were not fine-tuned for this task, we

Table VIII
EXPRIEMENT RESULTS FOR MODELS RUN ON NEWS STATEMENTS FEATURE ONLY

| Model | Features | Embedding | # Shots | Mean Accuracy | STD |
|---|---|---|---|---|---|
| SVM | News Statement | Tf-Idf | - | 22.34% | 0.23 |
| Random Forrest | News Statement | Tf-Idf | - | 23.44% | 0.42 |
| BiLSTM | News Statement | Linear | - | 20.54% | 0.26 |
| BiLSTM | News Statement | Word2Vec | - | 21.88% | 2.76 |
| TextCNN | News Statement | Linear | - | 21.14% | 0.79 |
| TextCNN | News Statement | Word2Vec | - | 25.24% | 0.46 |
| **Gecko** | **News Statement** | **Gecko** | **-** | **30.42%** | **0.34** |
| Voyage | News Statement | Voyage | - | 28.16% | 0.40 |
| LLama | News Statement | - | Zero | 20.50% | - |
| GPT | News Statement | - | Zero | 26.70% | - |
| Gemini | News Statement | - | Zero | 20.99% | - |
| LLama | News Statement | - | One | 19.90% | - |
| GPT | News Statement | - | One | 25.30% | - |
| Gemini | News Statement | - | One | 20.84% | - |
| LLama | News Statement | - | Two | 13.60% | - |
| GPT | News Statement | - | Two | 24.30% | - |
| Gemini | News Statement | - | Two | 21.23% | - |

Table IX
RESULTS FOR MODELS TRAINED ON NEWS STATEMENT AND TEXT CONTEXT. THE BEST MODEL IS HIGHLIGHTED IN BOLD

| Model | Features | Embedding | # Shots | Mean Accuracy | STD |
|---|---|---|---|---|---|
| SVM | New Statement + Text Context | Tf-Idf | - | 25.97% | 0.12 |
| Random Forrest | New Statement + Text Context | Tf-Idf | - | 25.40% | 0.92 |
| BiLSTM | New Statement + Text Context | Linear | - | 19.91% | 1.09 |
| TextCNN | New Statement + Text Context | Linear | - | 20.60% | 0.78 |
| Gecko | New Statement + Text Context | Gecko Single embedding for both Features | - | 26.16% | 1.43 |
| **Gecko** | **New Statement + Text Context** | **Gecko Separate Embedding for each feature** | **-** | **27.52%** | **1.02** |

Table X
RESULTS FOR MODELS TRAINED ON NEWS STATEMENT AND TEXT CONTEXT. THE BEST MODEL IS HIGHLIGHTED IN BOLD

| Model | Features | Embedding | # Shots | Mean Accuracy | STD |
|---|---|---|---|---|---|
| SVM | New Statement + Text Context + Speaker History | Tf-Idf | - | 35.7% | 0.45 |
| Random Forrest | New Statement + Text Context + Speaker History | Tf-Idf | - | 37.6% | 0.24 |
| BiLSTM + Fuzzy CNNBiLSTM | New Statement + Text Context + Speaker History | Linear | - | 37.34% | 2.07 |
| TextCNN + Fuzzy CNNBiLSTM | New Statement + Text Context + Speaker History | Linear | - | 40.69% | 1.69 |
| Gecko + Fuzzy Linear | New Statement + Text Context + Speaker History | Gecko separate embedding for both Features | - | 42.84 | 0.656 |
| **Gecko +Fuzzy** | **New Statement + Text Context + Speaker History** | **Gecko separate Embedding for each feature** | **-** | **44%** | **1.19** |

believe fine-tuning these models for this specific task should increase performance.

– **Transfer Learning:** LIAR dataset has only 12,836 data points. Pretraining the models on multiple fake news detection tasks before fine-tuning on LIAR dataset may reduce overfitting and improve performance.
– **Emphasising Interpretability and Explainability:** While in this research we experimented with the effect of different features as inputs to the model and their effect on performance, further investigation into why these features improve performance for some models and reduce it for others is yet to be investigated.

## References


[1] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Information sciences*, vol. 497, pp. 38–55, 2019.

[2] F. Farhangian, R. M. Cruz, and G. D. Cavalcanti, "Fake news detection: Taxonomy and comparative study," *Information Fusion*, vol. 103, p. 102140, 2024.

[3] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD explorations newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[4] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of economic perspectives*, vol. 31, no. 2, pp. 211–236, 2017.

[5] X. Zhou, R. Zafarani, K. Shu, and H. Liu, "Fake news: Fundamental theories, detection strategies and challenges," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 836–837.

[6] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *Acm Computing Surveys (Csur)*, vol. 51, no. 2, pp. 1–36, 2018.

[7] C. Kang and A. Goldman, "In washington pizzeria attack, fake news brought real guns," *New York Times*, vol. 5, p. A1, 2016.

[8] C. Dewey, "Facebook has repeatedly trended fake news since firing its human editors," *Washington Post*, vol. 12, 2016.

[9] V. L. Rubin, "Deception detection and rumor debunking for social media," in *The SAGE handbook of social media research methods*. Sage London, 2017, p. 342.

[10] B. Edkins, "Americans believe they can detect fake news. studies show they can't," 2016.

[11] C. Domonoske, "Students have 'dismaying'inability to tell fake news from real, study finds," *National Public Radio*, vol. 23, 2016.

[12] J. Lee, Z. Dai, X. Ren, B. Chen, D. Cer, J. R. Cole, K. Hui, M. Boratko, R. Kapadia, W. Ding *et al.*, "Gecko: Versatile text embeddings distilled from large language models," *arXiv preprint arXiv:2403.20327*, 2024.

[13] J. Alghamdi, S. Luo, and Y. Lin, "A comprehensive survey on machine learning approaches for fake news detection," *Multimedia Tools and Applications*, vol. 83, no. 17, pp. 51 009–51 067, 2024.

[14] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[15] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] C.-O. Truică and E.-S. Apostol, "It's all in the embedding! fake news detection using document embeddings," *Mathematics*, vol. 11, no. 3, p. 508, 2023.

[17] H. Reddy, N. Raj, M. Gala, and A. Basava, "Text-mining-based fake news detection using ensemble methods," *International journal of automation and computing*, vol. 17, no. 2, pp. 210–221, 2020.

[18] G. McIntire, "Fake and real news dataset," *GeorgeMcIntire's Github*, 2018.

[19] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big data*, vol. 8, no. 3, pp. 171–188, 2020.

[20] W. Y. Wang, "" liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

[21] C. Liu, X. Wu, M. Yu, G. Li, J. Jiang, W. Huang, and X. Lu, "A two-stage model based on bert for short fake news detection," in *Knowledge Science, Engineering and Management: 12th International Conference, KSEM 2019, Athens, Greece, August 28–30, 2019, Proceedings, Part II 12*. Springer, 2019, pp. 172–183.

[22] C. Xu and M.-T. Kechadi, "An enhanced fake news detection system with fuzzy deep-learning," *IEEE Access*, 2024.

[23] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[24] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," *Advances in neural information processing systems*, vol. 32, 2019.

[25] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8,

p. 9, 2019.

[26] T.-H. Cheung and K.-M. Lam, "Factllama: Optimizing instruction-following language models with external knowledge for automated fact-checking," in *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2023, pp. 846–853.

[27] "Excited to announce voyage embeddings!" https://blog.voyageai.com/2023/10/29/voyage-embeddings/, accessed: 2010-09-30.

[28] D. Boissonneault and E. Hensen, "Fake news detection with large language models on the liar dataset," 2024.

[29] B. Upadhayay and V. Behzadan, "Sentimental liar: Extended corpus and deep learning models for fake claim classification," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.

[30] A. Vlachos and S. Riedel, "Fact checking: Task definition and dataset construction," in *Proceedings of the ACL 2014 workshop on language technologies and computational social science*, 2014, pp. 18–22.

[31] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.

[32] G. K. Shahi, A. Dirkson, and T. A. Majchrzak, "An exploratory study of covid-19 misinformation on twitter," *Online social networks and media*, vol. 22, p. 100104, 2021.

[33] T. Mikolov, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, vol. 3781, 2013.