**Need to assign:**

* **Abstract**
* **Conclusion**
* **Reference**
* **Proposed Model (Framework) Architecture:**

**INCLUDE EACH TEAM MEMBERS CONTRIBUTION FOR THE PROJECT.**

# **Abstract**

# 

# 

# 

# 

# 

# 

# **1. Introduction**

This project proposes a deep-learning approach to address the issue of detecting fake news. The necessity to distinguish between real news and fake news is vital these days where information spreads quickly through social media and other online platforms. The significance of this matter extends to real-world implications such as defending the public against false information that could cause panic, damage reputations, influence elections, and destroy the people's faith in media sources.

Contemporary research utilizes either a single dataset or, if multiple datasets are employed, they are applied to distinct machine learning or deep learning models individually. Further, these datasets typically originate from a specific source or context, which may lead to issues with the model's fairness as the model trained on a singular or source-specific dataset may inadvertently inherit biases inherent to those sources. Augmenting data from two different sources may mitigate this issue by diversifying the dataset, which may help the model to generalize better and foster greater fairness in its decision-making. In addition, much of the research that has been done on Fake News Detection is based on traditional machine learning models such as Logistic Regression, Random Forest, XGBoost, etc. The latest research has used transformer-based models like BERT (Bidirectional Encoder Representations from Transformers), yielding good results. This study explores, implements, and evaluates Fake News Detection's performance on the newest State-of-the-Art Large Language Model - Meta’s Llama-2 7B transformer. As Llama-2 7B is trained on one trillion tokens, it is expected to capture vast amounts of contextual information, allowing it to understand the meaning of words and phrases in context. This can improve the model's ability to detect subtle nuances and recognize linguistic patterns associated with fake news.

# **2. Related Work**

Bharadwaj, et al. (2023) used the false and real news dataset, which includes more than 40,000 articles with fake and real news, to address the growing problem of fabricated news, particularly in the context of social media and the internet's widespread effect. In order to achieve the highest level of accuracy in false news identification, the authors conducted an in-depth evaluation and contrast of many different deep learning models, including Lightning Module, Logistic Regression, LSTM, Word Embedding, RNN, and Bag of Ngrams. The study concluded that RNN emerged as the more efficient model due to its ability to handle sequential input data.

Jaiswal et al. (2023) implemented RoBerta and Firefly optimization for feature extraction and selection. They used ISOT and FakeNewsNet datasets for their research. The optimized feature vector was fed into - Bi-LSTM, VGGNet, and CNN-supervised deep-learning-based models for classification. Bi-LSTM performed the best on both datasets with 75.90% accuracy and 76.77% F1-score on ISOT, and 86.30% accuracy and 87.60% F1-score on FakeNewsNet.

In the 2023 paper by T. Mahara et al. (2023), an in-depth analysis was conducted using the Fake News Healthcare dataset, which consists of 9,581 articles. The study implemented a Decision Tree, Random Forest, Support Vector Machine, AdaBoost-Decision Tree, and AdaBoost-Random Forest, Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) and Convolutional Neural Network-Bidirectional Long Short-Term Memory (CNN-BiLSTM). The results indicated that the AdaBoost-Random Forest model achieved the highest F1 score of 98.9%, closely followed by the CNN-LSTM model, which garnered an F1 score of 97.09%.

Rana et al. (2023) investigated the efficacy of pre-trained distilled BERT model versions to separate online rumors from fake news using the Fake News Challenge dataset. To determine how effectively the distilled BERT-tiny and BERT-small models can learn the main properties necessary to differentiate between fake and authentic news, researchers examined the outcomes obtained using different methods before and after integrating two smaller pre-trained BERT models into one framework. The findings reveal that BERT-small achieved 90.11% accuracy and demonstrated that these more compact variations were equally accurate to the related research while remaining effective, compact, and simple to train.

In their 2022 paper, Mahara and Gangele (2022) conducted an extensive study utilizing a publicly available dataset from Kaggle, which comprised 37,000 instances. The researchers employed two types of recurrent neural network models for their analysis: Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (Bi-LSTM). Their empirical evaluation revealed that the Bi-LSTM model outperformed the traditional LSTM model in accuracy with a score of 94%.

The study led by Chen et al. (2021) focused on the COVID-19 fake news dataset to investigate the effectiveness of various machine learning models in identifying misinformation. The models evaluated in the study included BERT (Bidirectional Encoder Representations from Transformers), RoBERTa (Robustly Optimized BERT), ALBERT (A Lite BERT), COVID-TWITTER-BERT (CT-BERT), and a specialized model termed Robust-COVID-Twitter-BERT (Ro-CT-BERT). Ro-CT-BERT emerged as the most effective model, achieving an accuracy rate of 99%.

Xavier et al. (2021) categorized, classified, and detected Fake News on Online Social Media (OSM) space. They identified seven types of fake news in OSM networks but worked on false connections between title and content and fabricated content. They employed the Stance model and trained Logistic Regression, Decision Tree, Random Forest, Multinomial Naive Bayes, and SVM Classifier to find false connections. Logistic Regression performed the best with 90.3% accuracy. For the fabricated content classifier, LSTM and Bi-LSTM were implemented where Bi-LSTM outperforms all other models by yielding 93.4% accuracy. The solution can be strengthened in the future by creating more machine-learning models or methods for identifying other sorts of fake news.

Shu et al. (2020) introduces a comprehensive data repository called FakeNewsNet with three features: news content which is the textual characteristics of the news, social context which is the social reactions to the news, and spatiotemporal information which provides details about when and where the articles were posted or shared (Shu et al., 2020). Support vector machine, Naïve Bayes, logistic regression, CNN, and social article fusion models (SAF, SAF/A, SAF/S) are developed to classify news based on their content and for social context evaluation, the paper uses Social article fusion model. For PolitiFact, the best model is Social Article Fusion with 0.691 accuracy score and for GossipCop, the highest accuracy score is 0.723 with CNN model (Shu et al., 2020).

In their paper, Hiramath & Deshpande (2019) proposed a fake news detection system based on classification algorithms such as Logistic regression, Support vector machine, Naïve Bayes algorithm, Random Forest algorithm, and Deep neural networks. Stemming and stop word removal were used on the News dataset as part of data pre-processing and utilized the Backpropagation algorithm to get the training data. As a result, they conclude that DNN performs better in terms of execution time and accuracy of 91% but requires more memory than other methods.

Thota et al. (2018) propose neural network architecture to predict fake news by classifying the relationship between articles’ title and content into ‘agree’, ‘disagree’, ‘discuss’ or ‘unrelated’. The project uses Fake News Challenge dataset and implements Tf-Idf Vectors with Dense Neural Network (DNN), Bag of Words (BOW) Vector with Dense Neural Network, and Pre-trained word embeddings with Neural Networks models. As a result, Tf-IDF - DNN outperforms other models with an accuracy score of 0.94 after being tuned and BOW - DNN is the second-best model (Thota et al., 2018).

# **3. Methodology**

Based on an extensive literature review, the project plans to implement three deep learning methods for fake news binary classification: Deep Neural Networks (DNN), Bidirectional Long Short-Term Memory (Bi-LSTM), and Meta’s Llama-2 7B transformer.

These three models have been chosen for their complementary strengths. DNNs are effective for fake news classification because they capture complex language patterns and nuances. Their scalability makes them well-suited for large datasets, and their relative explainability is beneficial for understanding the model's predictions. Bi-LSTM models understand words' sequence and context, making them ideal for analyzing the relationships within news articles. Their ability to capture past and future contextual information provides a more comprehensive understanding of the text, which is crucial for classifying fake news. Llama 2 offers a more advanced approach by focusing on the relationships between text parts through its novel attention method called grouped-query attention (GQA) and its fine-tuning mechanisms called Ghost Attention(GAtt) and Reinforcement Learning from Human Feedback (RLHF), which should allow for state-of-the-art (SOTA) performance for this classification task.

Before training these three models, preprocessing will be conducted. Each model will differ slightly in approach to how preprocessing will be conducted. For example, the DNN will make use of TF-IDF to improve explainability, Bi-LSTM will make use of word embeddings (Word2Vec or GloVe) to better capture word relationships, Llama 2 will use the Llama tokenizer, which is based on byte-level Byte-Pair-Encoding. The preprocessing includes text cleaning to remove irrelevant characters, tokenization to break down the text into smaller pieces, stopword removal to eliminate common but uninformative words, stemming or lemmatization to reduce words to their root form, word embedding to convert tokens into numerical vectors, and sequence padding to standardize the length of text sequences.

## **Performance Evaluation**

As this project addresses the binary classification problem, three evaluation metrics such as accuracy, F1-score, and Confusion Matrix have been selected to assess and compare the model's performance. Due to the ease of interpretation and straightforwardness accuracy is considered one of the measures, whereas an F1- score is a well-suitable measure for imbalanced datasets. Considering these two metrics is important to make decisions about model selection and fine-tuning. Based on the findings, the ranking of performance is as follows: Llama 2, Bi-LSTM, and DNN with Llama 2 demonstrating high performance. Also to compare the model results, DNN will be used as a baseline model without any hyperparameter tuning.

# **4. Dataset Description**

**ISOT Fake News dataset** from the University of Victoria.

* + <https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/>

The ISOT dataset encompasses articles categorized into two distinct types: fake and real news, primarily emphasizing topics related to politics and global affairs.

**Data Features:** Title, Text, Subject, Date

**FakeNewsNet**

* + <https://www.kaggle.com/datasets/mdepak/fakenewsnet>

The FakeNewsNet repository comprises datasets from both Buzzfeed News and Politifact, each containing distinct sets of articles categorized as either real or fake news.

**Data Features:** id, title, text, url, top\_img, authors, source, publish\_date, movies, images, canonical\_link, meta\_data

| **Datasets** (number of rows from different sources) | **ISOT** | **Politifact** | **Buzzfeed** |
| --- | --- | --- | --- |
| **Real** | 21418 | 123 | 92 |
| **Fake** | 23503 | 124 | 92 |

# 

# **5. Proposed Model (Framework) Architecture**

# **6. Experimental Setup:**

## **6.1 Data Cleaning:**

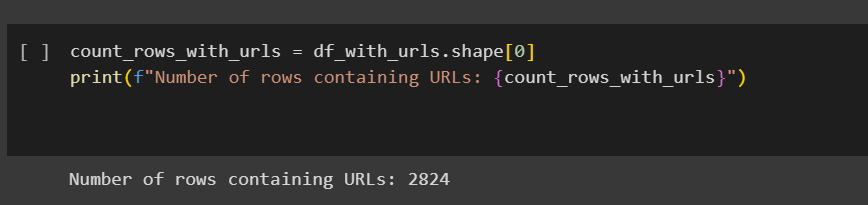
### Data EDA + Merging - Vidushi

### Data Cleaning for NLP -Joshna, Vidushi

Once the datasets were combined, the team conducted Exploratory Data Analysis (EDA) to gain a deeper insight into the raw data. Before initiating the modeling phase, it was crucial to preprocess the data. As a preliminary step, an examination was conducted to identify rows containing URLs starting with "https" and "www." This analysis revealed that 2,824 rows, as illustrated in Figure XX below, contained text with URLs. These URLs were subsequently extracted from the text using Python's "re" module, which effectively matched and removed the specified patterns in each row.

**Figure XX**

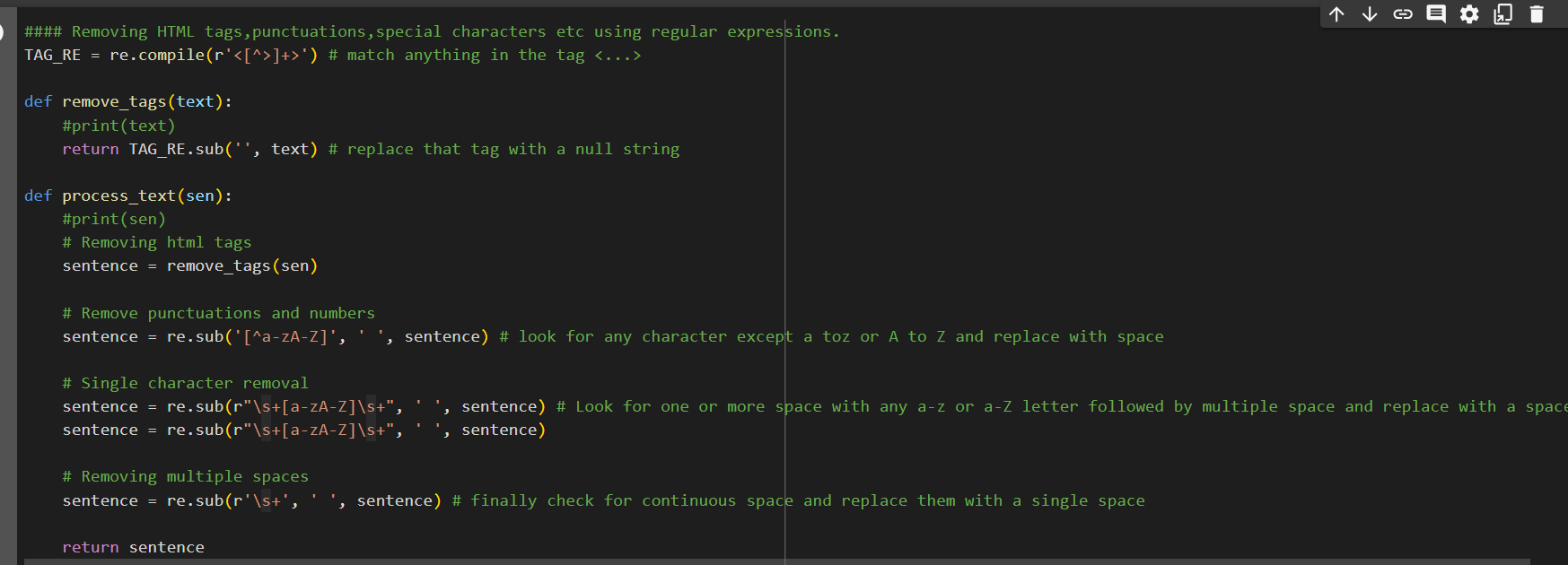
*Count of Rows containing URLs*



The second step in our data preprocessing involved checking for and removing email addresses within the text. This was accomplished using the `re` module in Python, which effectively identified and eliminated email addresses. Additionally, the preprocessing phase included the removal of HTML tags, punctuations, and special characters. This was achieved using a custom-defined function, which also leveraged the `re` module. The process and its results are depicted in Figure XX below.

**Figure XX**

*User Defined function for preprocessing*

**

After completing the initial cleaning, the team proceeded with NLP-based preprocessing on the data. This involved several steps: tokenizing the 'text' column into individual words, converting all words to lowercase, removing stopwords to filter out irrelevant words, and finally, performing lemmatization. Lemmatization was chosen over stemming as it takes into account the context of a word, transforming it into its more meaningful base form. All these tasks were carried out using the NLTK library in Python, a powerful tool for natural language processing.

### Data Augmentation -Vidushi

## **6.2 Data Preprocessing:**

### ***6.2.1 DNN: Nghi***

### ***6.2.2 Bi-LSTM: Vidushi***

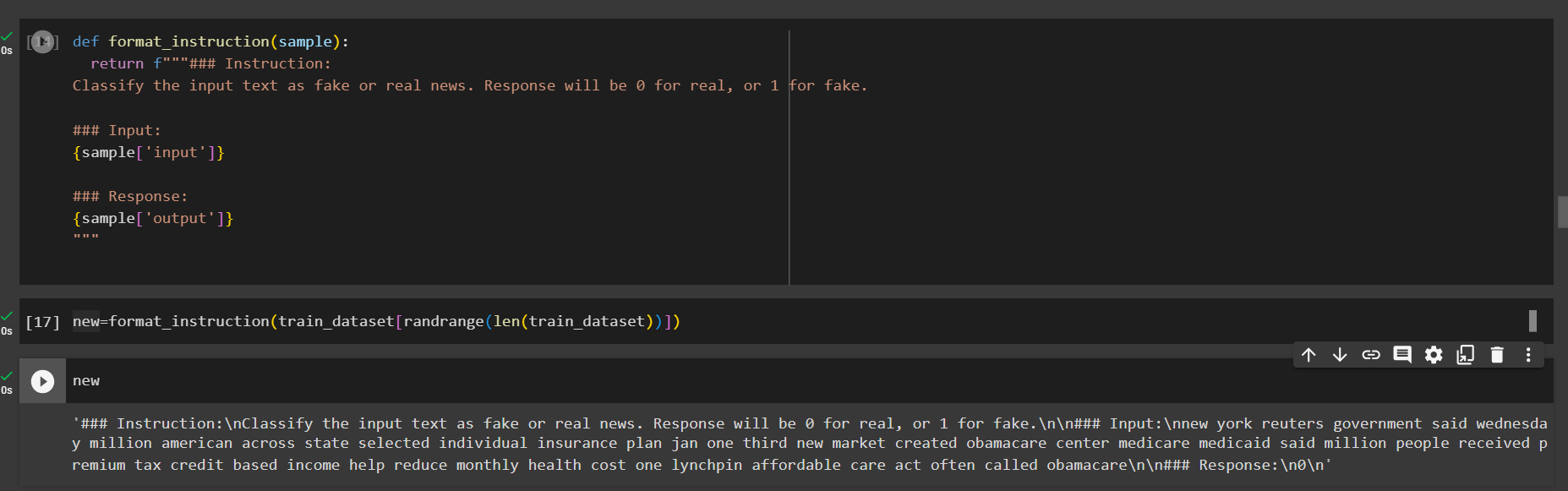
### ***6.2.3 LSTM and Logistic Regression: Sangamithra***

### ***6.2.4 Meta Llama 2 7B: Edward, Joshna***

Although the dataset underwent NLP-based preprocessing, further transformation was necessary to make it compatible with the LLaMA2 - 7B model. The dataset's 'text' column was renamed to 'input', and the 'target' column to 'output'. It was then split into 80% training and 20% testing subsets. A key aspect of the transformation process was the development of a prompt template. This template integrated an instructional statement that clearly defined the task, along with the corresponding value from the 'input' column. The expected model response was articulated, correlating with the data in the 'output' column. A sample prompt, generated from a random training dataset instance, is showcased in Figure XXX.

**Figure XX**

*Prompt template sample*

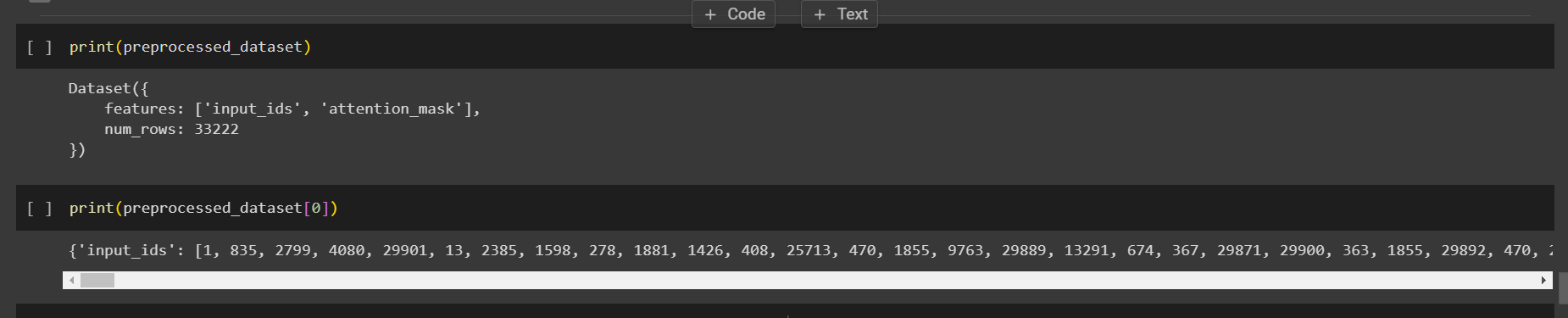
**

While the preprocessing steps for both training and test sets are largely similar, there is a subtle difference in the prompt format. Specifically, the test set does not include the Response value in its prompts. This is because the Response value represents the target output that the model is expected to predict, and is thus omitted in the test scenarios.

Once the training set's prompt template was established, the resultant formatted text was saved into a newly created 'formatted\_text' column. Subsequently, all other columns were removed from the dataset, leaving a singular column for batch processing. The dataset's batches are set with a maximum length of 1024 tokens, despite LLaMA2's capability of handling 4096, to accommodate the memory limitations of Colab. A custom preprocessing function was crafted to carry out these tasks, producing 'input\_ids' and 'attention\_masks' as outputs. The preprocessed dataset, detailed in Figure XX below, consists of 'input\_ids' — indices representing tokens transformed into a numerical format for model training, using a tokenizer from the Hugging Face Transformers library. The tokenizer is initialized with the configuration from the `bitsandbytes` model. The `bitsandbytes` library facilitates model quantization, a process that shrinks the size of deep learning models by decreasing the bit-width of the weights and activations. By employing quantization, models benefit from quicker inference times and lower memory usage, which is especially advantageous for running the models on edge devices with constrained computational capacity. Meanwhile, 'attention\_mask' is a binary sequence that discerns significant tokens from padding, ensuring the model focuses on the relevant data.

**Figure XX**

*Preprocessed dataset structure*

**

## **6.3 Model Development**

### ***6.3.1 DNN: Nghi***

### ***6.3.2 Bi-LSTM: Vidushi***

### ***6.3.3 LSTM, Logistic Regression: Mithra***

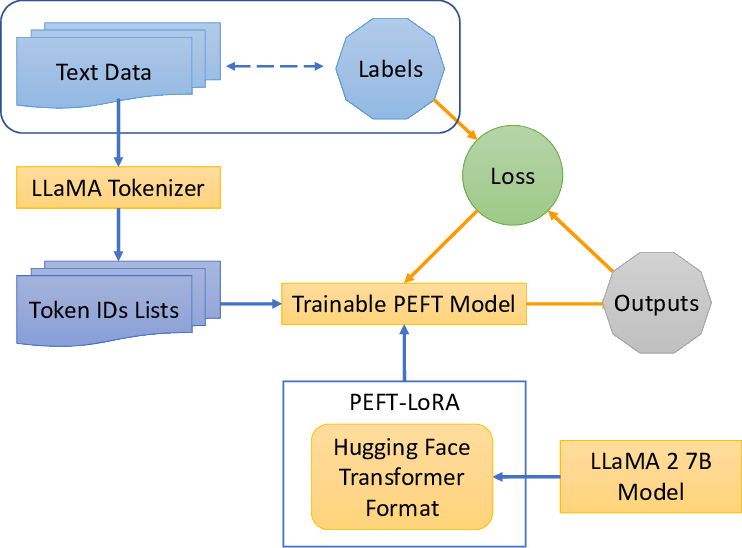
### ***6.3.4 Meta Llama 2 Edward Joshna:***

# Meta Llama 2 7B (Joshna)

LLaMA 2 represents a series of advanced pre-trained and fine-tuned large language models (LLMs), with sizes ranging from 7 billion to 70 billion parameters. Building on the foundation of the original LLaMA, LLaMA 2 utilizes the Google transformer architecture and introduces several enhancements. These enhancements include RMSNorm pre-normalization, akin to GPT-3's approach; the SwiGLU activation function, derived from Google's PaLM; the adoption of multi-query attention over the conventional multi-head attention; and the implementation of rotary positional embeddings (RoPE), similar to GPT Neo. The training of LLaMA utilized the AdamW optimizer for improved weight updates. Distinct from its predecessor, LLaMA 2 features a longer context window of 4096 tokens compared to 2048 and employs grouped-query attention (GQA) in its larger models, replacing the multi-query attention (MQA) mechanism (Heller, 2023). The data flow for the LLaMA2-7B model is illustrated in Figure XXX below, detailing the process from loading the model to tokenizing data, and finally, generating the loss value during training.

Figure XX

Training Data flow



To fine-tune the model, this project employed QLORA, an effective technique that quantizes a pretrained large language model (LLM) to 4 bits and integrates "Low-Rank Adapters," allowing for fine-tuning on a single GPU. This method is facilitated by the PEFT library. Users are required to log in to Hugging Face using the command `!huggingface-cli login` and an access token. A function `create\_bnb\_config` was defined to establish the `bitsandbytes` configuration before loading the model. The `bitsandbytes` library supports model quantization, and the `BitsAndBytesConfig` class from the `transformers` library is used to set up the quantization method. The pretrained model "meta-llama/Llama-2-7b-hf" was loaded with this configuration. Table XX below outlines the parameters selected for model loading via the bitsandbytes configuration.

**Table XX**

*BitsandBytes configuration parameters*

| BitsAndBytesConfig() | | |
| --- | --- | --- |
| load\_in\_4bit | True | Activate 4-bit precision base model loading |
| bnb\_4bit\_use\_double\_quant | True | Activate nested quantization for 4-bit base models (double quantization) |
| bnb\_4bit\_quant\_type | "nf4" | Quantization type (fp4 or nf4) |
| bnb\_4bit\_compute\_dtype | torch.bfloat16 | Compute data type for 4-bit base models |

After loading the model, as previously mentioned, we establish the Parameter-Efficient Fine-Tuning (PEFT) configuration specifically for the LoRA modules. This approach focuses on updating only a select subset of the model's parameters, greatly enhancing efficiency. With this PEFT configuration applied, the model is now primed and ready to commence training. The table XX below outlines a range of hyperparameters utilized in the fine-tuning process.

**Table XX**

*Hyperparametrs used for Fine-tuning*

| QLoRA parameters | |
| --- | --- |
| lora\_r | 8 |
| lora\_alpha | 64 |
| lora\_dropout | 0.1 |
| task\_type | "CAUSAL\_LM" |
| TrainingArguments parameters | |
| per\_device\_train\_batch\_size | 1 |
| learning\_rate | 5e-4 |
| optim | "paged\_adamw\_32bit" |
| gradient\_accumulation\_steps | 8 |

**Model Evaluation**

***DNN With Active Learning→ Nghi***

***Bi-LSTM → Vidushi***

***LSTM and Logistic Regression → Mithra***

***Meta Llama 2 7B With Active Learning → Edward and Joshna***

# **7. Results and Analysis**

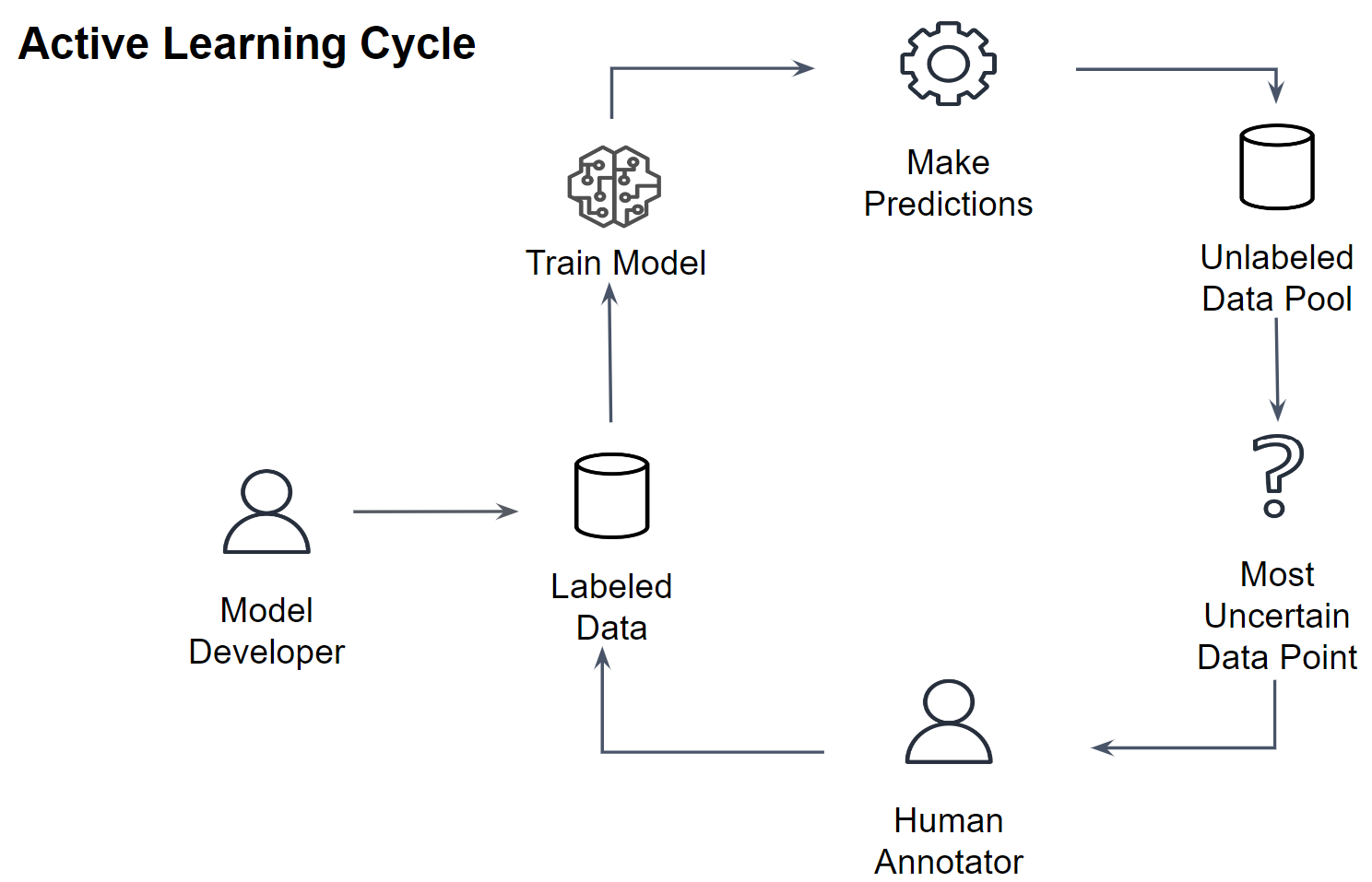
# CNN+Bi-LSTM Active Learning - (Edward)

~~The preprocessing stage of the CNN+Bi-LSTM active learning pipeline begins with transforming the target variable into a numerical format using Label Encoding, a common practice in handling categorical data for classification tasks. This is followed by a strategic division of the dataset into 70% training, 15% validation, and 15% test sets, ensuring that the model is trained, validated, and tested on distinct subsets of data. A crucial step in processing the textual data involves tokenizing the text using Keras's Tokenizer and padding the sequences to a uniform length. This standard procedure allows for efficient batch processing in neural networks. Additionally, the implementation utilizes pre-trained Word2Vec embeddings from Google (google-news-300) to convert text tokens into dense vectors. This technique effectively captures the semantic meanings of words, which is particularly beneficial in NLP.~~

~~In the methodology for the CNN+Bi-LSTM, the model employs an active learning approach, a semi-supervised technique where the model iteratively queries for labels on new data points. Initially, the model trains on a small set of data with labels, in this case, 20 samples. It progressively queries the most informative samples from the unlabelled pool, determined by their uncertainty levels. This process involves training the model in each iteration and making predictions on the unlabeled pool, using entropy as a criterion for sample selection. The most uncertain samples are added to the training set one at a time with each query and provided their respective label. An early stopping mechanism based on validation accuracy is incorporated to prevent overfitting, ensuring that the model training halts when no significant improvement in accuracy is observed after three epochs (Settles, 2009).~~

**~~Figure~~** ~~\_~~

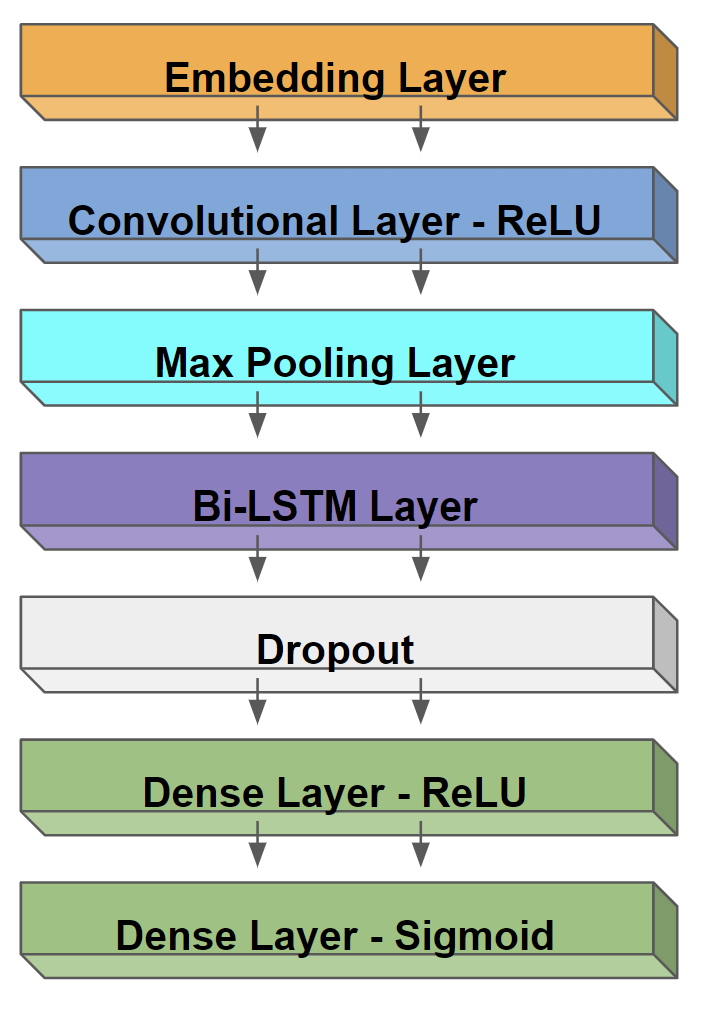
*~~Diagram of Active Learning Cycle~~*



~~The model's architecture is a hybrid of Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM), a combination that offers comprehensive feature extraction for text classification tasks such as Fake News detection. The CNN layer is adept at extracting features, such as n-grams, from the text, while the Bi-LSTM layer excels in capturing long-range dependencies by processing data in both forward and backward directions. This blend allows the model to grasp the textual features and the broader context effectively. The architecture is further enhanced with Dropout and Dense layers, providing the necessary regularization and classification capabilities, making it a suitable choice for complex text-based binary classification tasks. The figure below depicts the model architecture design.~~

**Figure** \_

*CNN+Bi-LSTM Architecture*



The experimental setup for the CNN+Bi-LSTM model is characterized by first conducting hyperparameter tuning on the training data to find the best combination of learning rate, and batch. This tuning process process produced the best hyperparameters for the learning rate as 0.001, and batch size 32. The Adam optimizer was chosen for its effectiveness in handling sparse gradients and its adaptive learning rate adjustment, which negates the need to manually set a learning rate. The model is trained for a maximum of 10 epochs in each iteration of the active learning loop. An epoch corresponds to one complete pass through the entire training dataset. This number is a standard choice, aiming to provide sufficient training time without excessive overfitting with the addition of early stopping on the validation set. The model utilizes relu (Rectified Linear Unit) activation functions in the Convolutional layer and the first Dense layer. Relu is a popular choice for deep learning models due to its ability to handle non-linear data effectively and its computational efficiency. The final output layer uses a sigmoid activation function, which is appropriate for binary classification tasks, as it squashes the output to a probability between 0 and 1, in this case real or fake news. In total, the model has 7 layers, if we count the dropout layer.

The number of neurons can be counted as 128 neurons in the convolutional layer, 128 in the bidirectional LSTM layer (64 in each direction), and 64 in the first dense layer, culminating in a single neuron in the output layer. The model has 3,259,265 total parameters, of which 2,929,265 are trainable, meaning these parameters will be updated during the training process, and 3,260,000 are non-trainable, corresponding to the fixed Google Word2Vec embeddings.

The hardware used to train the model was an A-100 GPU from Google Colab Pro +. The runtime to conduct the testing of 30 random\_state interactions took around 4 hours to complete.

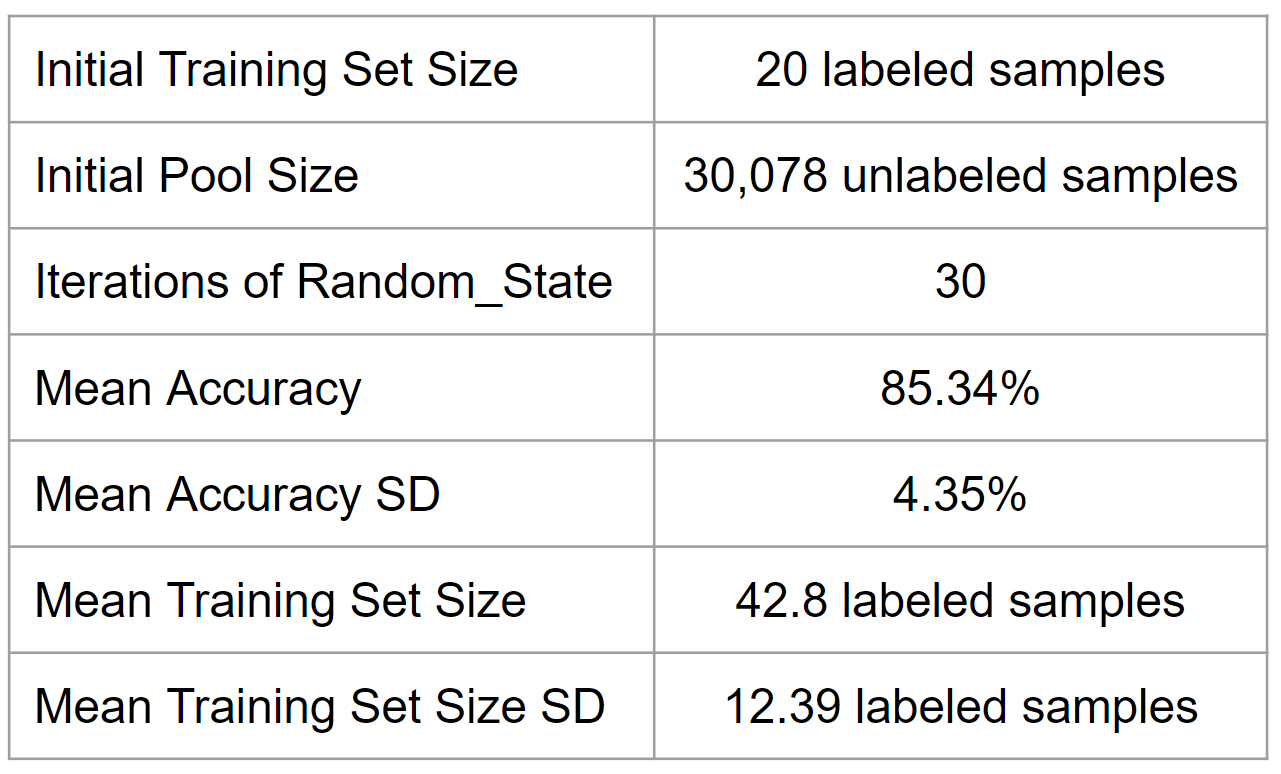
The model was developed using TensorFlow and Keras. TensorFlow provides a robust foundation for building and training models, while Keras, acting as a high-level API for TensorFlow, allows for streamlined and accessible model development. The exact versions of these libraries are critical for ensuring consistent performance, which were version 2.14.0 for both.

~~Accuracy is the chosen metric for performance evaluation. The model undergoes 30 runs with varying random states, a strategy that helps assess its stability and performance under different conditions, similar to that of cross-validation. The active learning component includes specific parameters like the number of queries, initial training size, patience, and tolerance, all of which contribute to the model's learning process. Finally, the model's effectiveness is evaluated on a test set post the active learning loop, providing insights into its generalization capabilities.~~

~~In evaluating the model's performance across 30 iterations with varying `random\_state` values, several key insights emerge that reflect the model's effectiveness and reliability. The mean accuracy achieved by the model is 85.34%, which is quite impressive, indicating a high level of proficiency in classifying the text data on just 42.8 samples with labels. This level of accuracy, especially in a complex task like text classification, demonstrates the model's robustness and the efficacy of the combined CNN-BiLSTM architecture along with the active learning approach. However, the standard deviation of 04.35% in accuracy across different iterations suggests some variability in performance. This variability could be attributed to the tolerance threshold on the validation accuracy to prevent the model from overfitting as it continues performing queries. The table below shows the model performance on the test data over the span of 30 iterations of the random\_state variable.~~

**Table** \_

*CNN+Bi-LSTM Results on Test Data for 30 Iterations*



~~Another important aspect is the mean training set size, which stands at 42.8 samples with a standard deviation of 12.39 samples. This result is particularly interesting as it starts from an initial size of just 20 samples. The increased training set size due to active learning illustrates the model’s capability to query and learn from the most informative samples selectively. The standard deviation in training set size indicates variability in the number of queries needed across different iterations to reach optimal performance. This is a natural outcome of the active learning process where each iteration might converge at different rates based on the initial data subset.~~

~~Overall, these results underscore the model's success in adapting and learning from a dynamically changing training set, achieved through the active learning framework. The high mean accuracy points to the model's effectiveness in Fake News text classification, while the standard deviation in both accuracy and training set size highlights the impact of initial data selection and the inherent uncertainty in the learning process. This analysis suggests that while the model is generally effective, there could be room for further optimization, particularly in reducing performance variability across iterations.~~

# **8. Conclusion + future work?**

# **References**

Heller, M. (2023, September 12). *What is Llama 2? Meta’s large language model explained*. InfoWorld. <https://www.infoworld.com/article/3706470/what-is-llama-2-metas-large-language-model-explained.html#:~:text=Llama%202%20is%20a%20family,fine%2Dtuned%20for%20programming%20tasks>.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. Retrieved from http://burrsettles.com/pub/settles.activelearning.pdf