Name:	**Anushka Sanjay Thakur**
College Name:	Veermata Jijabai Technological Institute, Matunga, Mumbai (Maharashtra-400019)
Mentor:	Mr. Sudheerkumar Y
Topic:	**Fake News detection using Deep Learning**
Problem Statement:	**Fake News Detection Model on Social Media by Leveraging Sentiment Analysis of News Content and Emotion Analysis of Users' Comments**

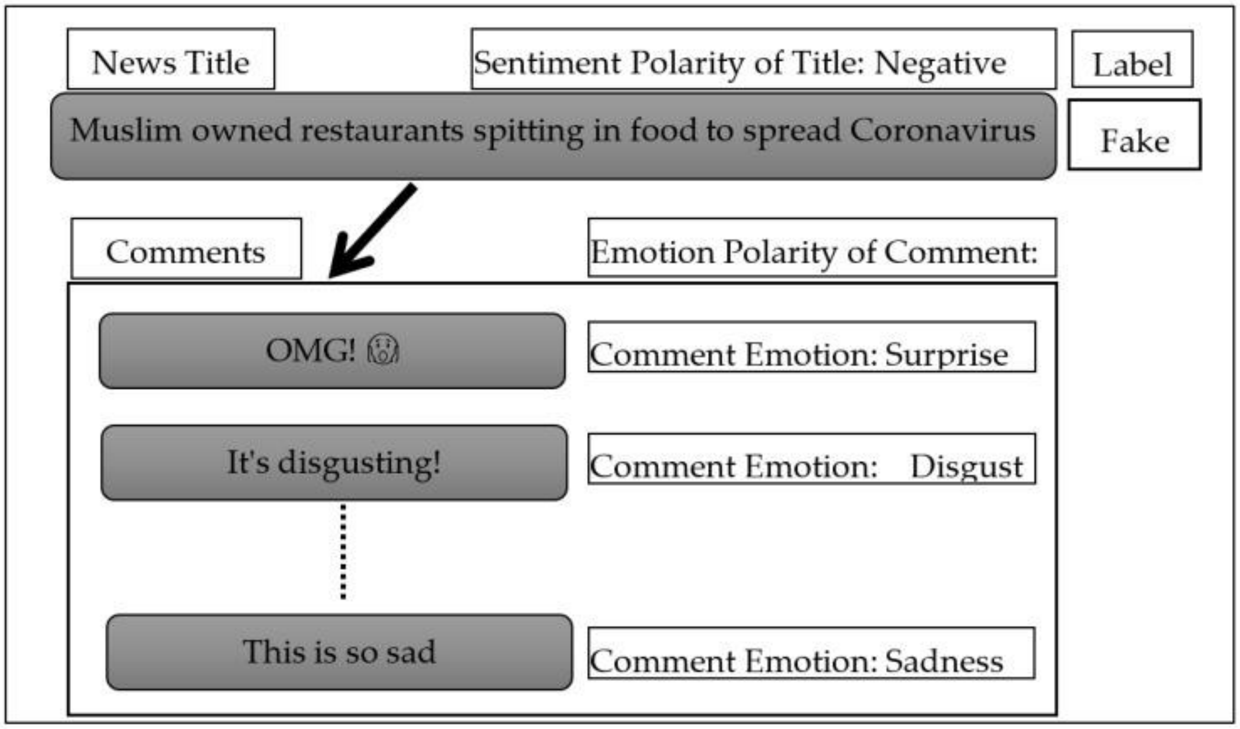
Fake News Detection Model on Social Media by Leveraging Sentiment Analysis of News Content and Emotion Analysis of Users' Comments

In today's world, social media has become the primary source of news for many people. The widespread dissemination of fake news on social networks has become a significant global problem, negatively impacting various aspects of society, including politics, economics, and social well-being. Fake news often conveys negative sentiments, and the public's reaction to it can be characterized by emotions such as surprise, fear, and disgust.
Keywords: fake news detection, sentiment analysis, emotion analysis, social media, deep learning

1. Introduction

Fake news affects people's daily lives, manipulates their thoughts and feelings, changes their beliefs, and may lead them to make wrong decisions. The propagation of fake news on social media negatively affects society in many domains such as political, economic, social,

health, technological, and sports. Social media platforms have presented a virtual environment for posting, discussion, exchange of views, and global interaction among users, without restrictions on location, time, or content volume. In 2021, Facebook announced that about 1.3 billion fake accounts had been closed, and more than 12 million posts containing false information about COVID-19 and vaccines had been deleted.
The current research proposes a model using a new approach that employs new trends in the detection of fake news on social networks by utilizing features extracted based on sentiment and emotion analysis, taking care not to neglect the emojis in the comments, as shown in figure below, which is a real example of fake news (https://thelogicalindian.com/fact-check/muslim-spit-restaurant-covid-19-coronavirus-20457,) based on a report conducted by the BBC (https://www.bbc.com/news/world-asia-india-53165436. This fake news about the Muslim community in India was disseminated in the first week of April 2020; it claimed that Muslims deliberately spread COVID-19. This led to increased hostilities toward Muslims and calls to boycott them economically. The importance of the research lies in the fact that most of the studies that dealt with the matter of emotion analysis-based features of the public's responses in detecting fake news only provided analysis and statistics of the news in the dataset in terms of containing certain emotions, and these features were not tested and practically utilized in detecting fake news.


News Title
Sentiment Polarity of Title: Negative
Label
Muslim owned restaurants spitting in food to spread Coronavirus
Fake
Comments
Emotion Polarity of Comment:
OMG! 😱
Comment Emotion: Surprise
It's disgusting!
Comment Emotion: Disgust
This is so sad
Comment Emotion: Sadness

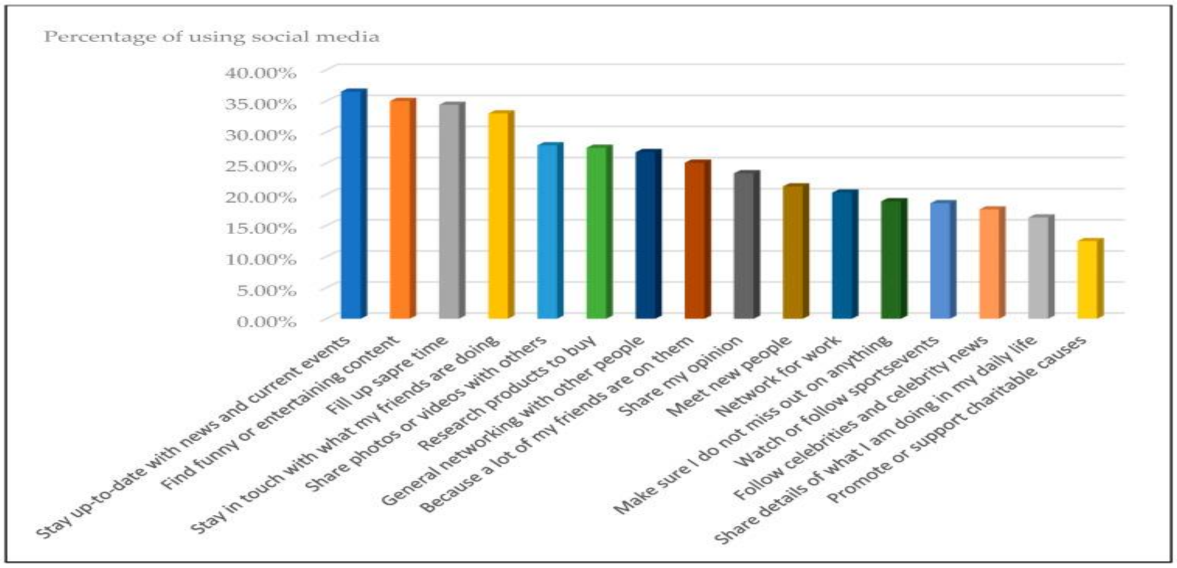
My contribution to the project is as follows:

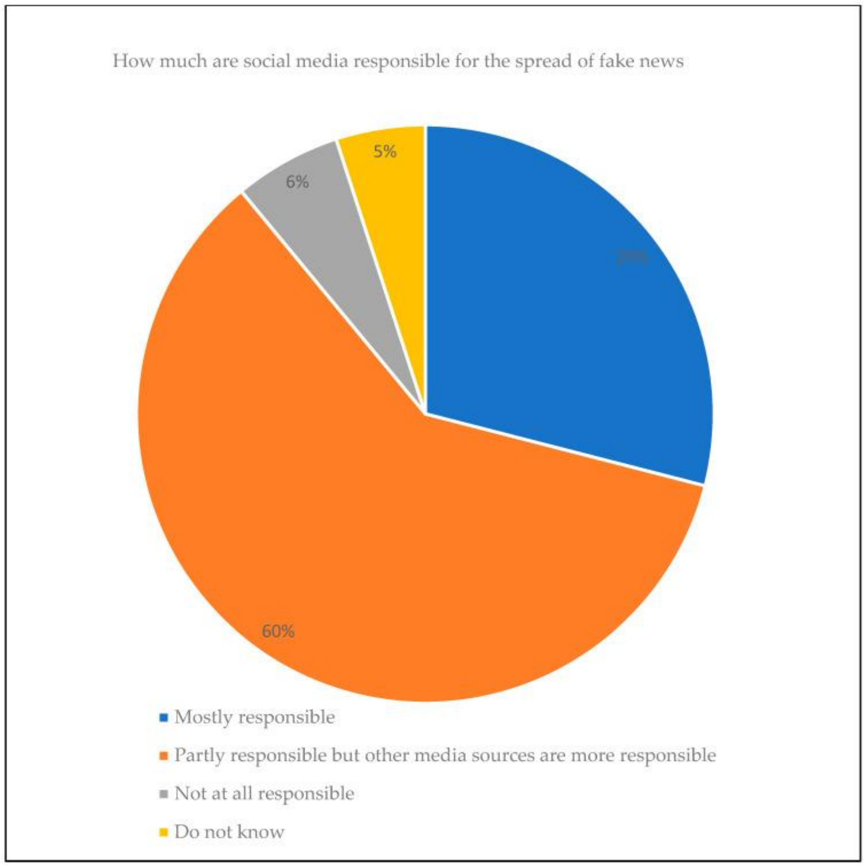
- Analyzing the sentiments of news titles and also analyzing the emotions of users' comments on this news and their relationship to fake news in the Fakeddit dataset.
- Examine the textual content of news headlines in detecting fake news.

- Investigating the use of sentiment-based features of news headlines, as well as features based on the emotions of users' comments, to detect fake news.

Details Related to Social Media Users	Number or Percentage
Total number of active social media users	4.20 Billion
Social media users as a percentage of the global population	53.60%
Annual change in the number of global social media users	+13.2% (+490 Million)
Total number of social medias accessing via mobile phones	4.15 Billion
Percentage of total social media users accessing via mobile	98.80%

The reason for using social media according to the Digital 2021 Global Digital Overview report



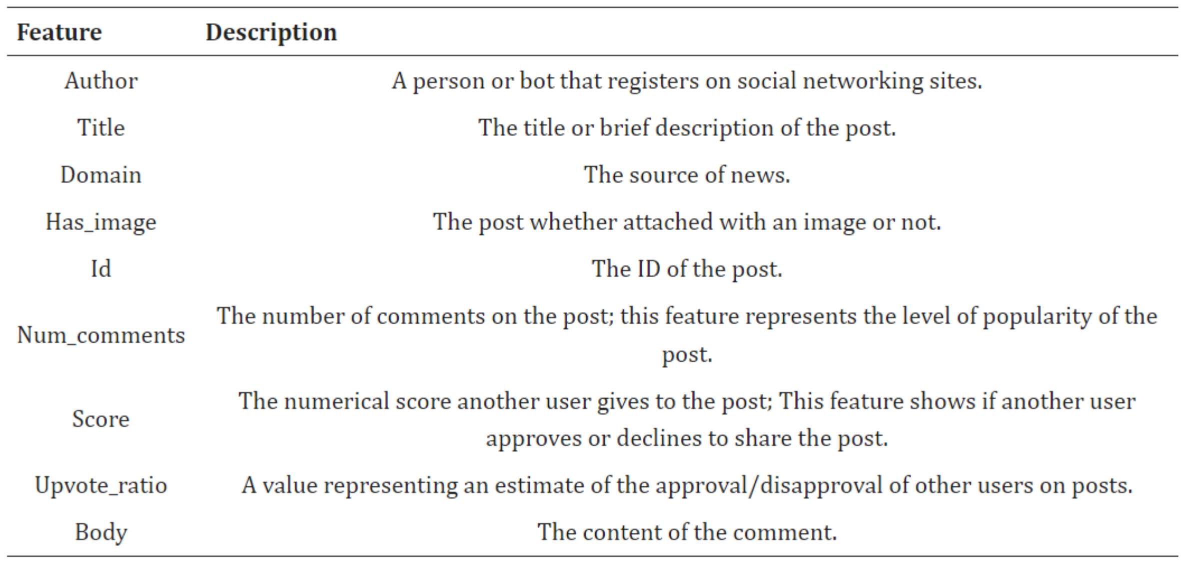


Social media’s role in spreading fake news (Available from: https://www.statista.com/statistics/649221/fake-news-expose-responsible-usa).

2.1.1. Dataset Description

The Fakeddit (https://github.com/entitize/Fakeddit, dataset is a large-scale and multi-modal dataset (text and image) collected from the social media Reddit platform for the period from 19 March 2008 to 24 October 2019, **by researchers Nakamura, Levy.** This dataset consists of more than one million posts from various domains. Several features are associated with these posts such as images, comments, users, domains, and other metadata. This dataset contains a lot of noise and null values. A single post may contain multiple comments and may not have a comment. For each post, the researchers provided three labels, classified in two ways, three ways, and six ways.

The description of the Fakeddit dataset.



Feature	Description
Author	A person or bot that registers on social networking sites.
Title	The title or brief description of the post.
Domain	The source of news.
Has_image	The post whether attached with an image or not.
Id	The ID of the post.
Num_comments	The number of comments on the post; this feature represents the level of popularity of the post.
Score	The numerical score another user gives to the post; This feature shows if another user approves or declines to share the post.
Upvote_ratio	A value representing an estimate of the approval/disapproval of other users on posts.
Body	The content of the comment.

3.1.1. Sentiments Analysis in Natural Language Processing (NLP)

Based on TextBlob (https://pypi.org/project/textblob/0.9.0/), which is a lexicon-based sentiment analyzer, the sentiment of the titles in this Fakeddit dataset was analyzed. Sentiment polarity is the output that TextBlob produces after receiving a title. The output represents the polarity that falls between (−1 and 1), where −1 indicates a negative sentiment and +1 indicates a positive sentiment, and between them, a value of zero denotes a neutral sentiment. The results of analyzing the news titles from the dataset used in this study, using the mentioned technique are consistent with what has been reported in previous studies, which showed that fake news titles tend to be negative more than positive, but real news titles tend to be positive, although most of the titles in both classes are moderate.

Data Preprocessing steps and workflow:

1. **Load the Dataset**:
    - The code loads the dataset files (multimodal_train.tsv, multimodal_validate.tsv, and multimodal_test_public.tsv) using the pd.read_csv() function.
    - It checks if the 'title' column exists in the dataset, as it is a crucial feature for the analysis.
2. **Sentiment Analysis**:
    - The code defines a function get_sentiment_polarity() that calculates the sentiment polarity of the text using the TextBlob library.
    - It applies this function to the 'title' column of the dataset to obtain the sentiment polarity scores for each title.

   - The sentiment polarity scores range from -1 (negative) to 1 (positive), with 0 representing neutral sentiment.
   - The code then analyzes the sentiment distribution by counting the number of positive, negative, and neutral samples.
   - The results are plotted using a bar chart to visualize the sentiment distribution.
3. **Sentiment Polarity Saving**:
   - The code saves the sentiment polarity scores for the 'title' column to a CSV file named 'sentiment_polarities.csv'.
4. **Text Preprocessing**:
   - The code defines several functions to preprocess the text:
     - remove_hash(): Removes the '' characters from the text.
     - remove_tag(): Removes the ' characters from the text.
     - remove_underscore(): Replaces underscores with spaces in the text.
   - It then applies these functions to the 'title' column of the dataset to clean the text.
5. **Stop-words Removal and Lemmatization**:
   - The code initializes the NLTK WordNetLemmatizer and the set of English stop-words.
   - It defines a function remove_stopwords_lem() that performs the following steps:
     - Tokenizes the text using word_tokenize().
     - Removes stop-words from the tokenized text.
     - Applies lemmatization to each word using the WordNetLemmatizer.
     - Joins the lemmatized words back into a single string.
   - The function is applied to the 'title' column of the train, validation, and test datasets, and the preprocessed texts are stored in separate lists.
6. **Tokenization and Padding**:
   - The code combines the preprocessed titles from the train, validation, and test datasets into a single list news_all.
   - It then creates a Tokenizer object and fits it on the news_all list to build the vocabulary.
   - The code tokenizes the preprocessed titles for the train, validation, and test datasets using the texts_to_sequences() method of the Tokenizer.
   - Finally, it pads or truncates the tokenized sequences to a fixed length of 15 using the pad_sequences() function.

<u>The rationale behind the implemented data preprocessing steps is to:</u>

1. **Sentiment Analysis**: Understand the sentiment distribution of the news titles, which can provide insights into the overall tone and emotional content of the dataset.
2. **Text Cleaning**: Remove noise and irrelevant characters from the text, such as punctuation, numbers, and multiple spaces, to improve the quality of the input data for the machine learning model.
3. **Stop-words Removal and Lemmatization**: Remove common stop-words (e.g., "the", "a", "and") that do not carry significant meaning, and perform lemmatization

to reduce words to their base forms, which can help the model better understand the semantic relationships between words.

4. **Tokenization and Padding**: Convert the preprocessed text into a numerical format that can be fed into the machine learning model. Padding the sequences to a fixed length ensures that all input samples have the same dimensionality, which is a requirement for many deep learning models.

The results of the data preprocessing steps are:

1. **Sentiment Polarity Scores**: The sentiment polarity scores for the news titles are saved to a CSV file, which can be used for further analysis or as an additional feature for the machine learning model.
2. **Preprocessed Text**: The cleaned and preprocessed text is stored in separate lists for the train, validation, and test datasets, ready to be used as input to the machine learning model.
3. **Tokenized and Padded Sequences**: The preprocessed text is tokenized and padded to a fixed length, creating numerical representations of the input data that can be used for training and evaluating the machine learning model.
4. **Model Building:**

The main goal of this project is to study both unimodal and multimodal approaches to deal with a finer-grained classification of fake news. To do this, I am using the Fakeddit dataset (Nakamura et al., 2020), made up of posts from Reddit. The posts were classified into the following six different classes: true, misleading content, manipulated content, false connection, imposter content and satire. I explore several deep learning architectures for text classification such as Convolutional neural network (CNN) (Goodfellow et al., 2016), Bidirectional long short-term memory (BiLSTM) (Hochreiter & Schmidhuber, 1997) and Bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018). As multimodal approach, I propose a CNN architecture that combines both texts and images to classify the fake news.

4.1. Unimodal approaches for fake news detection

I review the most recent studies for the detection of fake news using only the textual content of the news. **WanI et al. (2021) use the Constraint@AAAI Covid-19 fake news dataset (Patwa et al., 2020), which contains tweets classified as true or fake. Several methods are evaluated: CNN, LSTM, Bi-LSTM + Attention, Hierarchical Attention Network (HAN) (Yang et al., 2016), BERT, and DistilBERT (Sanh et al., 2019), a smaller version of BERT. The best accuracy obtained is 98.41 % by the DistilBERT model when it is pre-trained on a corpus of Covid-19 tweets**. GoldanI et al. (2021) use a capsule network model (Sabour et al., 2017) based on CNN and pre-trained word embeddings for fake news classification over the ISOT (Ahmed et al., 2017) and LIAR (Wang, 2017) datasets. The ISOT dataset is made up of fake and true news articles collected from Reuters and Kaggle, while the LIAR dataset contains short statements classified into the following six classes: pants-fire, false, barely-true, half-true, mostly-true and true. Thus, the authors perform both binary and multi-class fake news classification. The best accuracy obtained with the proposed model is 99.8 % for

the ISOT dataset (binary classification) and 39.5 % for the LIAR dataset (multi-class classification). Girgis et al. (2018) perform fake news classification using the above mentioned LIAR dataset. More concretely, they use three different models: vanilla Recurrent Neural Network (Aggarwal, 2018), Gated Recurrent Unit (GRU) (Chung et al., 2014) and LSTM. The GRU model obtains an accuracy of 21.7 %, slightly outperforming the LSTM (21.66 %) and the vanilla RNN (21.5 %) models.
From this review on approaches using only texts, we can conclude that deep learning architectures provide very high accuracy for the binary classification of fake news, however, the performance is much lower when these methods address a fine-grained classification of fake news. Curiously enough, although BERT is reaching state-of-the-art results in many text classification tasks, it has hardly ever used for the multiclassification of fake news.
(Reference-Google.com)

3. Methods

3.1. Unimodal approaches

Three unimodal models only using the texts are proposed: CNN, BiLSTM and BERT.

3.1.1. Preprocessing for deep learning models

We start by cleaning up the documents in the corpus. This includes removing common words (like "the" and "a"), punctuation, numbers, and extra spaces. Then, we break each text into smaller parts called tokens. Next, we apply lemmatization to the tokens. Lemmatization is a process that converts words to their base or dictionary form. For example, "running" would be converted to "run". After lemmatization, we transform the texts into sequences of numbers. We do this by first creating a vocabulary, which is a list of all the unique words in the corpus. Each word is then assigned a unique integer number. This creates a dictionary that maps words to numbers. Finally, we use this dictionary to convert each text into a sequence of integers. Each number in the sequence represents a word from the original text, and the order of the numbers reflects the order of the words in the original text.
Since deep learning models require input vectors of the same length, we need to pad and truncate the sequences of integers so that they all have the same number of entries. Padding means adding extra values (usually zeros) to the end of shorter sequences, while truncating means removing values from the end of longer sequences. The downside of this process is that for the sequences that are too long, some information will be lost when they are truncated. To determine the optimal length for the padded/truncated vectors, we calculated the percentage of texts in the corpus that have fewer than 10, 15, 20, and 25 tokens. This helps us understand the distribution of text lengths and select a vector length that balances retaining information and having consistent input sizes for the deep learning models. So in summary, we standardize the text input by padding and truncating the sequences of integers representing the words, and we use the text length statistics to choose the optimal vector length that preserves as much information as possible while meeting the requirements of the deep learning models.
Here I have explored two approaches for the word embeddings:

1. **Random initialization:** The word embeddings are initialized randomly and then further trained by the deep learning model.
2. **Pre-trained GloVe embeddings:** We use the GloVe word embeddings (Pennington et al., 2014), which are pre-trained on a large corpus of text. We can either keep these embeddings static (not train them further) or let the model fine-tune them (dynamic approach).

By using these approaches, we can investigate the impact of the word embeddings on the performance of the deep learning models.

3.1.2. CNN

The input to the CNN model is the matrix of word embeddings, which has a size of 15 rows (the maximum sequence length) and 300 columns (the dimensionality of the word embeddings). This matrix is passed through an embedding layer, which can be initialized using either random weights or pre-trained GloVe word embeddings. After the embedding layer, the model applies four different convolutional filters. These filters have sizes of (2 x 300), (3 x 300), (4 x 300), and (5 x 300), respectively. These filter sizes are commonly used in CNN-based text classification models, as they capture n-gram features of different lengths (Voita, 2021). As the filters slide across the input matrix, they generate 50 output channels for each filter size. This results in the following output shapes for the convolutional layer:

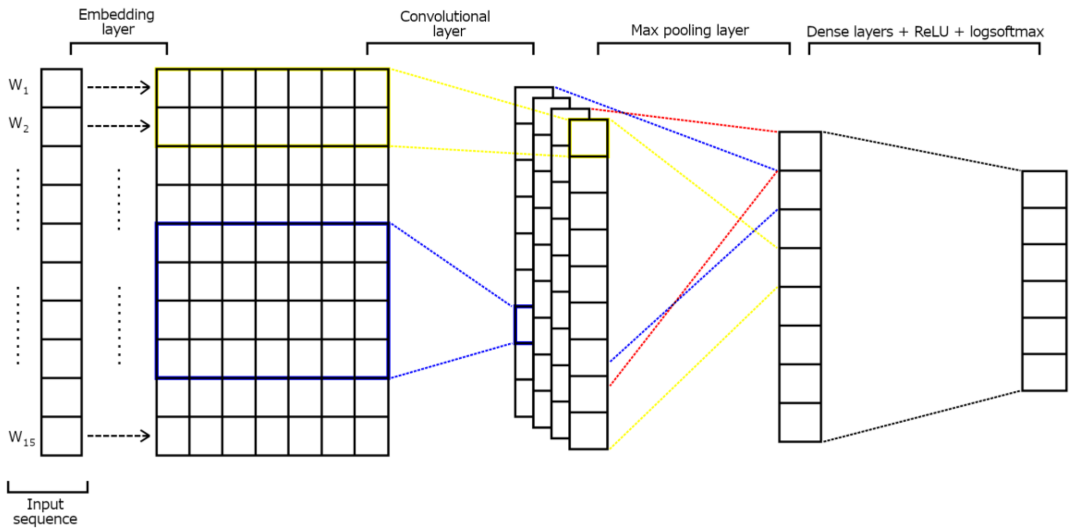
- **(2 x 300) filter: (14 x 1) output**
- **(3 x 300) filter: (13 x 1) output**
- **(4 x 300) filter: (12 x 1) output**
- **(5 x 300) filter: (11 x 1) output**

The rationale behind using multiple filter sizes is to capture different levels of contextual information in the text. The smaller filters (2 x 300 and 3 x 300) focus on local n-gram features, while the larger filters (4 x 300 and 5 x 300) capture longer-range dependencies in the text.

After the convolutional layer, the model applies pooling, typically max-pooling, to reduce the dimensionality of the features. This is followed by additional layers, such as fully connected layers and an output layer, to perform the final text classification task. The use of pre-trained word embeddings, such as GloVe, can help the model leverage the semantic information encoded in the embeddings, potentially improving the classification performance compared to randomly initialized embeddings.

After the convolutional layer, the output feature maps are passed through a ReLU (Rectified Linear Unit) activation function. This applies the element-wise function f(x) = max(0, x), setting all negative values to 0 while leaving positive values unchanged. Next, a max-pooling layer is applied to the ReLU-activated feature maps. This layer selects the maximum value from each of the 200 feature maps (50 feature maps per filter size), reducing the dimensionality of the features. The pooled features are then concatenated into a single vector and passed through two dense (fully connected) layers, with a ReLU activation in between.

The final output of the dense layers is a vector of six values, corresponding to the six classes in the Fakeddit dataset. This output is passed through a log-softmax function to obtain the predicted class probabilities. The model is trained using the Adam optimization algorithm and the negative log-likelihood loss function. Early stopping is employed to determine the appropriate number of training epochs, using the train and validation partitions.



Source: Google

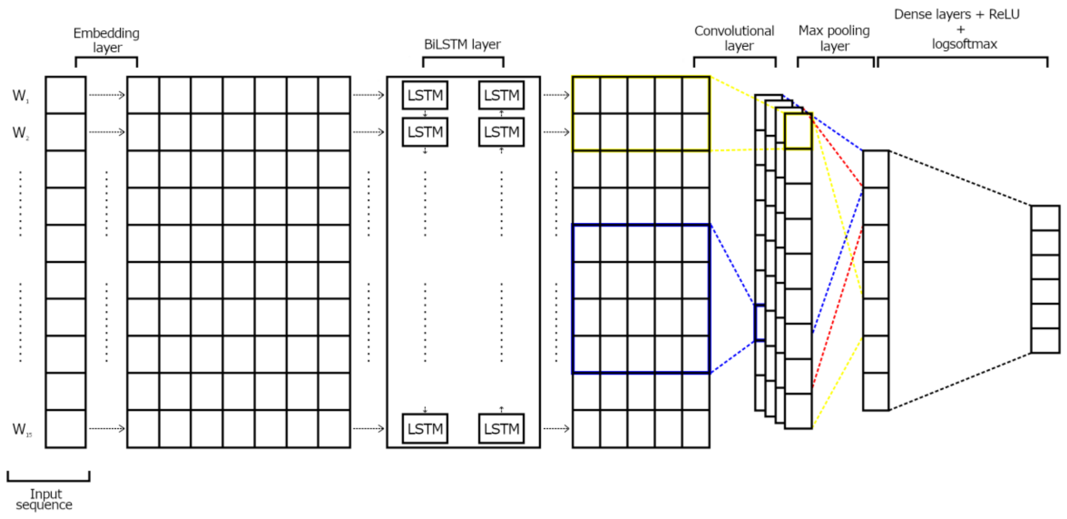
Figure 2 in the provided information shows the overall CNN architecture for the text classification task.

3.1.3. BiLSTM

This model is a hybrid model that combines a Bidirectional LSTM (BiLSTM) and a Convolutional Neural Network (CNN) for text classification. First, the input text is processed. Each input text is transformed into a vector of length 15, which is then passed through an embedding layer. This embedding layer converts each input vector into a matrix of size 15 x 300. This matrix represents the word embeddings for the input text. The word embedding matrix is then fed into a Bidirectional LSTM layer. A Bidirectional LSTM is a type of recurrent neural network that can capture information from both the forward and backward directions of the text. The BiLSTM layer has hidden states of length 70, and its output is a matrix of size 15 x 140, containing the hidden states from both the forward and backward directions.

The output of the BiLSTM layer is then passed through a convolutional layer. This layer applies 240 filters, each with a size of 3 x 140. This generates 240 output arrays, each with a size of 13 x 1. Next, a ReLU (Rectified Linear Unit) activation function is applied to the convolutional layer's output. This is followed by a max-pooling layer, which selects the largest element from each of the 240 feature maps. This results in a sequence of 240 numbers. The pooled features are then concatenated and passed through two dense (fully connected) layers with ReLU activation in between. Finally, the output of the dense layers is passed through a log-softmax function to obtain the predicted class probabilities.

The model is trained using the Adam optimization algorithm and the negative log-likelihood loss function. Early stopping is used to determine the optimal number of training epochs.



Source: Google

Figure 3 in the provided information illustrates the overall architecture of this BiLSTM-CNN hybrid model for text classification.

3.1.4. BERT

In this case, instead of using random initialization of the pre-traiend Glove embeddings, we now use the vectors provided by BERT to represent the input tokens. As opposed to the GloVe model (Pennington et al., 2014), BERT is taking into account the context of each word (that is, the words that surround it). 11 For the pre-processing of the texts the steps are similar to those described above. The main differences are that we tokenize the texts by using the BertTokenizer class from the transformers library (Face). This class has its own vocabulary with the mappings between words and ID's so it was not necessary to train a tokenizer with the corpus of texts. We also add the [CLS] and [SEP] tokens at the beginning and at the end of each tokenized sequence. It was also necessary to create an attention mask in order to distinguish what entries in each sequence correspond to real words in the input text and what entries are just 0's resulting from padding the sequences. Thus, the attention mask is composed of 1's (indicating non-padding entries) and 0's (indicating padding entries). We use the BERT base model in its uncased version (12 layers, 768 hidden size, 12 heads and 110 million parameters). Then, we fine-tune it on our particular problem, that is, the multi-classification of fake news. To do this, we add a linear layer on top of the output of BERT that receives a vector of length 768 and outputs a vector of length 6. For the training process, we used the Adam algorithm for optimization with a learning rate of $2·10−5$. We trained the model for two epochs, since the authors of BERT recommended using between two and four epochs for fine-tuning on a specific NLP task (Devlin et al., 2018).

3.2. Multimodal approach

Multimodal approach uses a CNN that takes as inputs both the text and the image corresponding to the same news. The model outputs a vector of six numbers out of which the predicted class is obtained. In the following lines, we describe the preprocessing steps applied before feeding the data into the network as well as the architecture of the network. Regarding the preprocessing of the images, we only reshaped them so that all have the same shape (560 x 560). Once the pre-processed data is fed into the network, different operations are applied to text and image. The CNN architecture that we use for the texts is the same that we described in 3.1.2, 12 except for the fact that we eliminate the last 2 dense layers with ReLU activation in between. We now describe the CNN model to classify the images. The data first goes through a convolutional layer. Since each image is made up three channels, the number of input cahnnels of this layer is also 3.

Besides, it has 6 output channels. Filters of size (5 x 5) are used with stride equal to 1 and no padding. The output for each input image is therefore a collection of 6 matrices of shape (556 x 556). **The output of the convolutional layer passes through a nonlinear activation function (ReLU) and then maxpooling is applied with a filter of size (2 x 2) and a stride equal to 2. The resulting output is a set of six matrices of shape (278 x 278).** The output from the maxpooling layer passes again through another convolutional layer that has 6 input channels and 3 output channels. The filter size, stride length and padding are the same as those used in the previous convolutional layer. Then the ReLU non-linear activation function and the maxpooling layer are applied again over the feature maps resulting from the convolutional layer. **Thus, for a given input (image) we obtain a set of 3 feature maps of shape (137 x 137).** Finally, these feature maps are flattened into a vector of length 56307. The outputs from the operations applied to each text and image are concatenated into a single vector. Then, this vector is passed through 2 dense layers with a ReLU non-linear activation in between. Finally, the log softmax function is applied and the logarithm of the probabilities is used in order to compute the predicted class of the given input.

Evaluation:

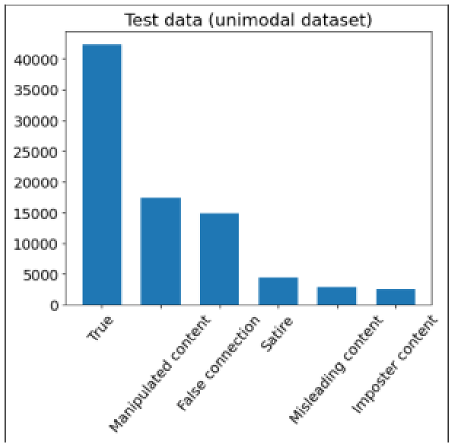
Dataset:

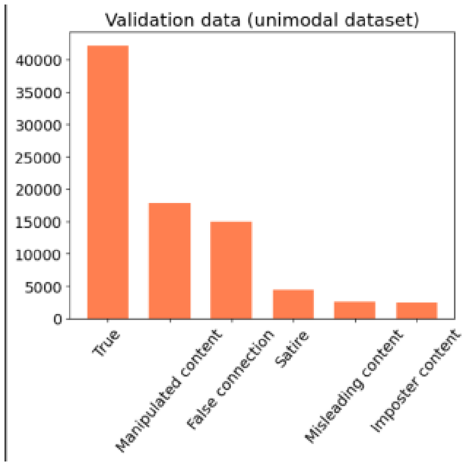
- Here the model building is done using the Fakeddit dataset, which contains over 1 million instances of posts and comments from Reddit.
- The Fakeddit dataset has the advantage of allowing for a more fine-grained classification of fake news, beyond just the binary true/fake distinction.
- Each instance in the dataset is labeled with one of 6 categories:
    - i. **True:** the news is true
    - ii. **Manipulated Content:** the content has been manipulated (e.g. photo editing)
    - iii. **False Connection:** the text and image are not aligned
    - iv. **Satire/Parody:** the content is twisted or misinterpreted in a satirical/humorous way
    - v. **Misleading Content:** the information has been deliberately manipulated to mislead

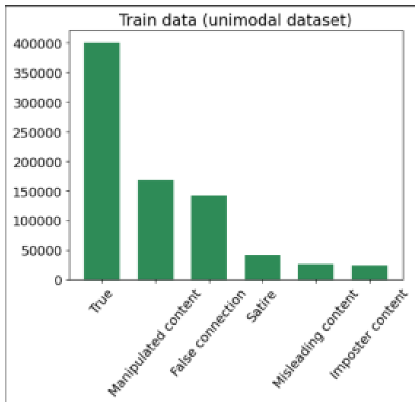
vi. **Imposter Content**: the content is generated by bots

In summary, the Fakeddit dataset provides a more nuanced set of labels for evaluating fake news detection models, going beyond just true vs. fake classification.

- The class distributions are similar across the unimodal and multimodal versions of the dataset, as well as across the training, validation, and test splits.
- Both the unimodal and multimodal datasets are imbalanced, with the "true", "manipulated content", and "false connection" classes having significantly more instances than the "satire", "misleading content", and "imposter content" classes.
- The imbalance in the dataset may make the classification task more challenging for the underrepresented classes (satire, misleading content, and imposter content), as there are fewer examples for the model to learn from.

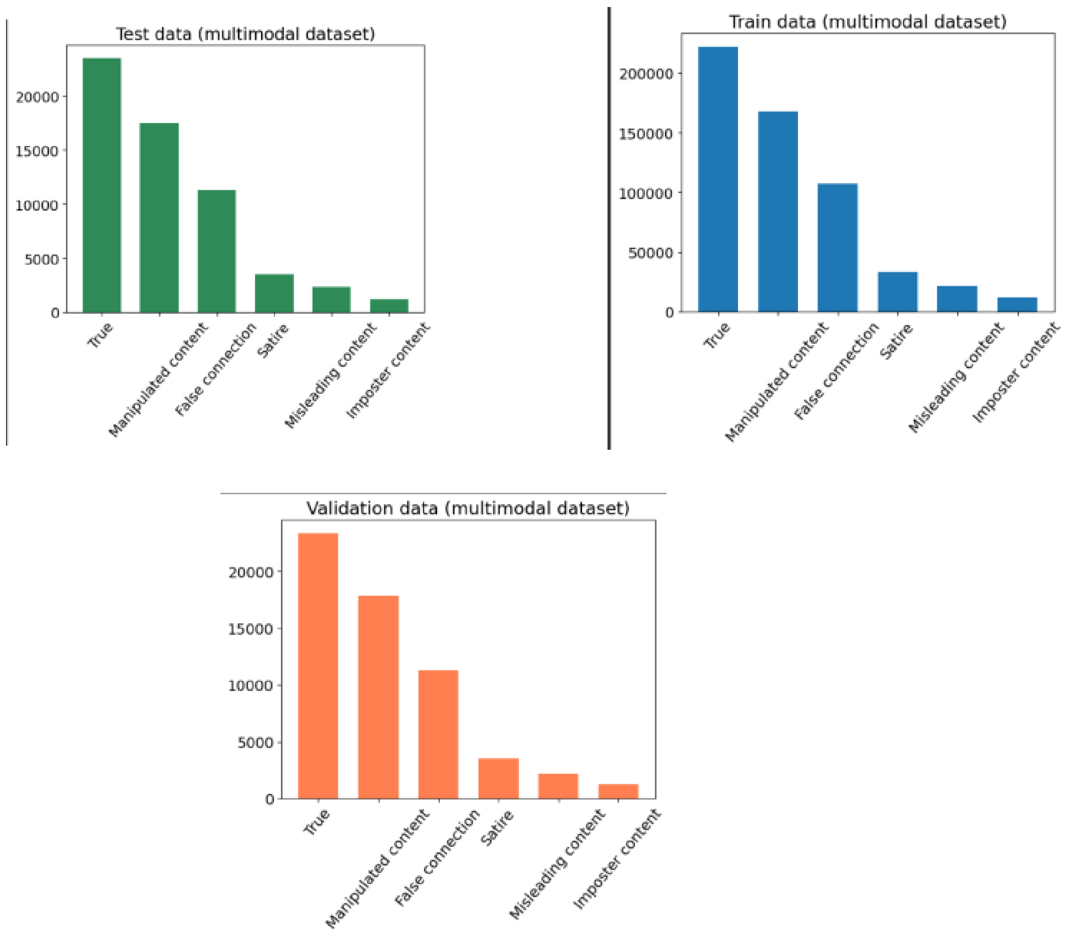






Source: Google Colab Notebook

Figure 4: Class distribution (unimodal dataset). Figure 4 shows the distribution of the classes in the unimodal dataset,



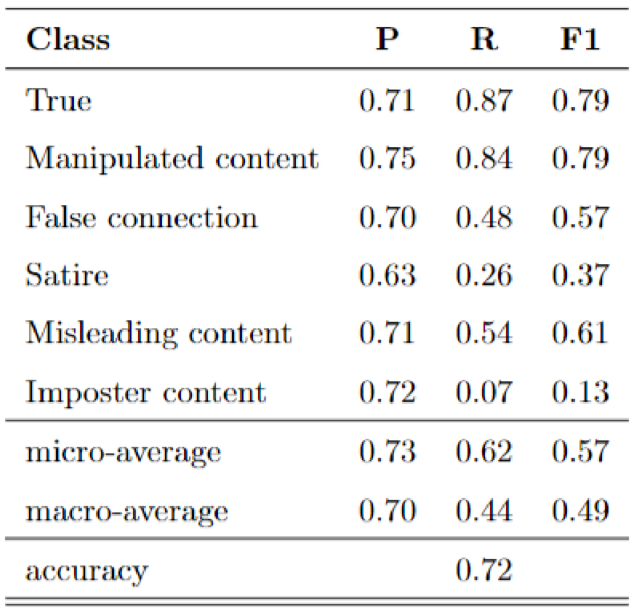
Source: Google Colab Notebook

Figure 5 shows the distribution for the multimodal dataset.

1. **The performance metrics reported for each model include**:
   - Recall, precision, and F1 score for each class
   - Overall accuracy across all classes
2. **The researchers are particularly interested in the models' ability to detect false content, so they also compute:**
   - Micro and macro averages of recall, precision, and F1 scores for the 5 fake news classes (excluding the "true" class)
   - F1 micro score and accuracy for the 5 fake news classes
3. The goal is to compare the performance of the different models and identify the best approach for detecting fake news, with a focus on how well the models can identify the various types of false content.
4. The use of these metrics, including the micro and macro averages for the fake news classes, allows the researchers to thoroughly evaluate and compare the models' capabilities in this fine-grained fake news classification task.

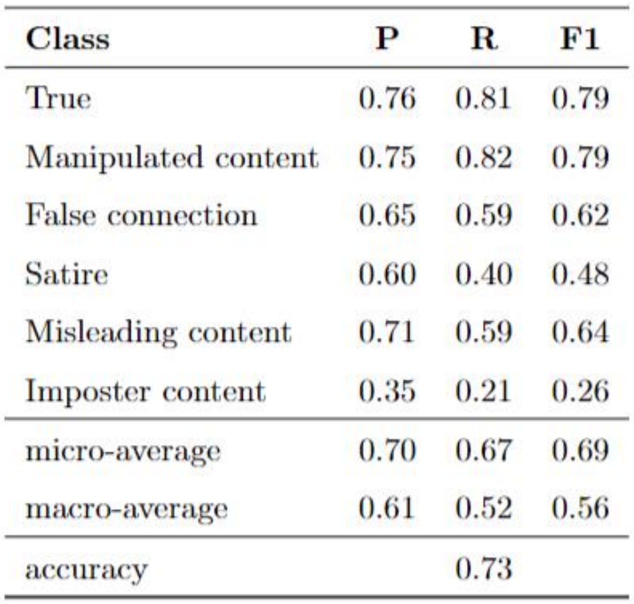
Results

In this section, we can present the results obtained for each model. We have report the recall, precision and F1 scores obtained by all the models for each class. The accuracy is computed over all the classes. It helps us to compare models and find the best approach. Moreover, we can say that we are also interested in knowing which model is better at detecting only those news containing false content. For this 16reason, we also compute the micro and macro averages of the recall, precision and F1 metrics only over five classes of fake news without the true news. We use the F 1micro score and the accuracy to compare the performance of the models

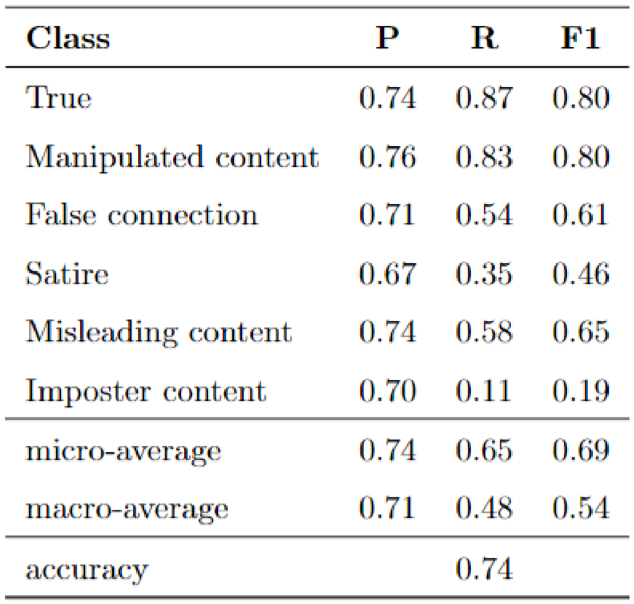


Class	P	R	F1
True	0.71	0.87	0.79
Manipulated content	0.75	0.84	0.79
False connection	0.70	0.48	0.57
Satire	0.63	0.26	0.37
Misleading content	0.71	0.54	0.61
Imposter content	0.72	0.07	0.13
micro-average	0.73	0.62	0.57
macro-average	0.70	0.44	0.49
accuracy		0.72	

In our initial experiment, the model with randomly initialized and updated embedding layer achieved 72% accuracy, 57% micro F1, and 49% macro F1. The True and Manipulated classes had the highest F1 (79%), while the minority Imposter class had the lowest (13%). However, the model performed well on the Misleading class (61% F1).
We then explored static and dynamic GloVe embeddings. The dynamic model had slightly higher accuracy, but the static model had a higher micro F1. Both achieved the same macro F1 of 69%. The main difference was in the Imposter class, where the dynamic model performed worse.



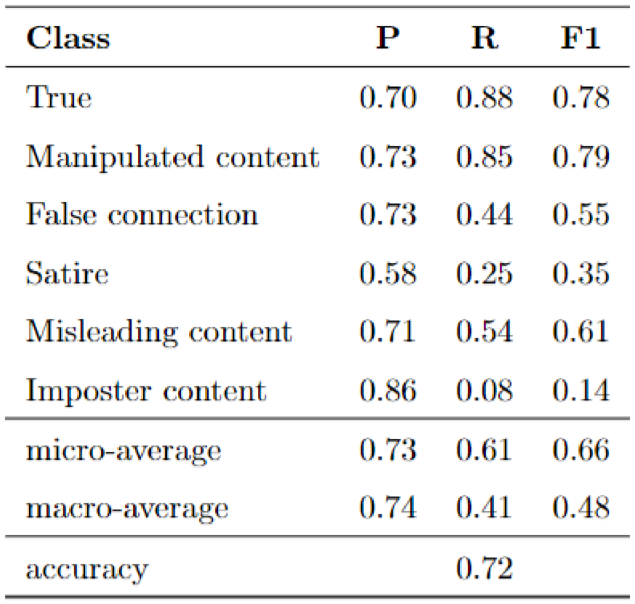
Class	P	R	F1
True	0.76	0.81	0.79
Manipulated content	0.75	0.82	0.79
False connection	0.65	0.59	0.62
Satire	0.60	0.40	0.48
Misleading content	0.71	0.59	0.64
Imposter content	0.35	0.21	0.26
micro-average	0.70	0.67	0.69
macro-average	0.61	0.52	0.56
accuracy		0.73	



Class	P	R	F1
True	0.74	0.87	0.80
Manipulated content	0.76	0.83	0.80
False connection	0.71	0.54	0.61
Satire	0.67	0.35	0.46
Misleading content	0.74	0.58	0.65
Imposter content	0.70	0.11	0.19
micro-average	0.74	0.65	0.69
macro-average	0.71	0.48	0.54
accuracy		0.74	

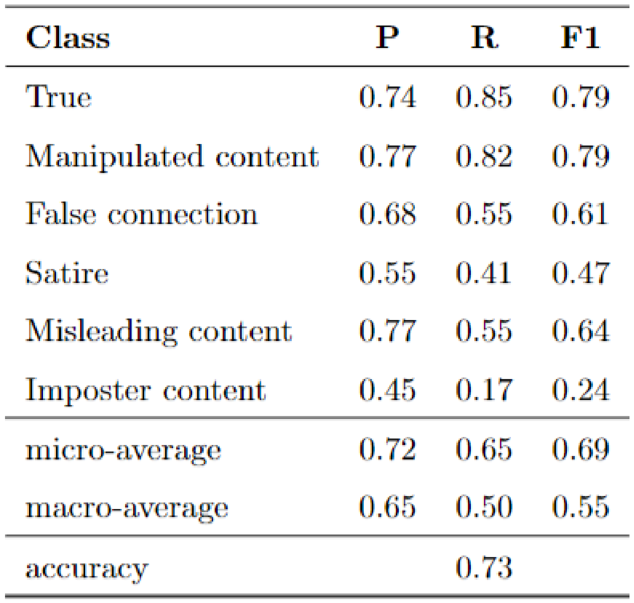
Using pre-trained GloVe embeddings, whether static or dynamic, outperformed random initialization, as the GloVe vectors capture meaningful word relationships.
Considering the imbalanced dataset, the micro F1 score showed the CNN with dynamic GloVe embeddings as the best-performing model.
However, the dynamic approach took considerably longer to train (6000-8000 seconds more) compared to the static GloVe model, due to the additional learning of word embeddings.

BiLSTM results.



Class	P	R	F1
True	0.70	0.88	0.78
Manipulated content	0.73	0.85	0.79
False connection	0.73	0.44	0.55
Satire	0.58	0.25	0.35
Misleading content	0.71	0.54	0.61
Imposter content	0.86	0.08	0.14
micro-average	0.73	0.61	0.66
macro-average	0.74	0.41	0.48
accuracy		0.72	

As a second deep learning model, we have explore a BiLSTM model. So, here I have replicate the same experiments as described for CNN, that is, using random initialization and pre-trained Glove vectors. The BiLSTM model with random initialization performed similarly to the CNN with random initialization in terms of accuracy, but had a 9-point higher micro F1 score, likely due to improved performance on the Imposter class.



Class	P	R	F1
True	0.74	0.85	0.79
Manipulated content	0.77	0.82	0.79
False connection	0.68	0.55	0.61
Satire	0.55	0.41	0.47
Misleading content	0.77	0.55	0.64
Imposter content	0.45	0.17	0.24
micro-average	0.72	0.65	0.69
macro-average	0.65	0.50	0.55
accuracy		0.73	

Using static GloVe embeddings significantly improved the BiLSTM's performance across several classes, such as a 6-point increase for False Connection, 12-point increase for Satire, and 10-point increase for Imposter, compared to random initialization.

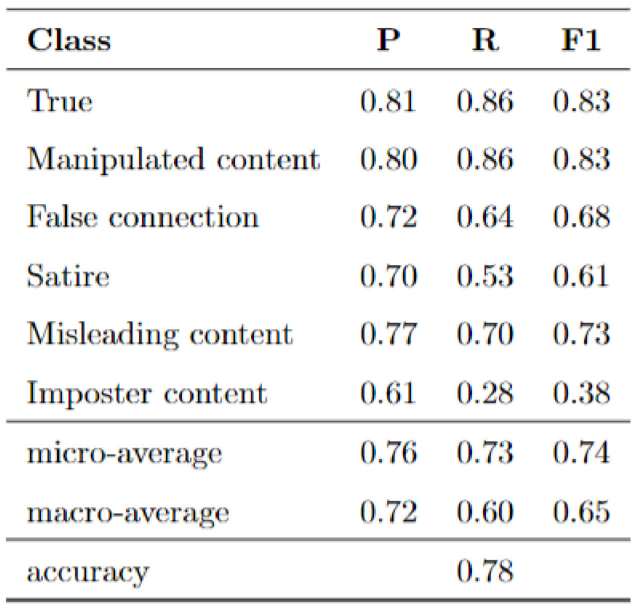
Updating the GloVe embeddings during training (dynamic approach) further boosted the BiLSTM's performance, achieving 75% accuracy, 2 points higher than the static GloVe BiLSTM. The dynamic GloVe BiLSTM also slightly outperformed the dynamic GloVe CNN model in micro F1.
However, the dynamic training approach took considerably more time than the static approach, due to the added complexity of learning the word embeddings along with the model parameters.

Class	P	R	F1
True	0.75	0.86	0.80
Manipulated content	0.77	0.84	0.80
False connection	0.72	0.55	0.63
Satire	0.63	0.41	0.50
Misleading content	0.78	0.57	0.66
Imposter content	0.57	0.18	0.28
micro-average	0.74	0.67	0.70
macro-average	0.69	0.51	0.57
accuracy		0.75	

BERT results.

Class	P	R	F1
True	0.81	0.86	0.83
Manipulated content	0.80	0.86	0.83
False connection	0.72	0.64	0.68
Satire	0.70	0.53	0.61
Misleading content	0.77	0.70	0.73
Imposter content	0.61	0.28	0.38
micro-average	0.76	0.73	0.74
macro-average	0.72	0.60	0.65
accuracy		0.78	

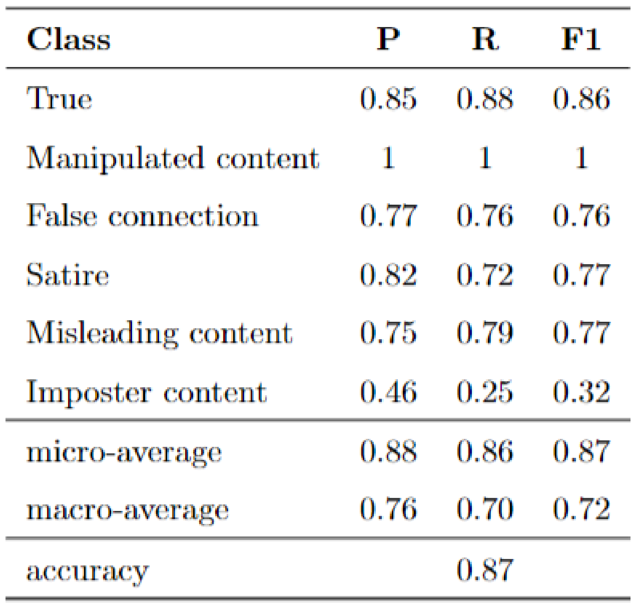
Unimodal Approach:



Class	P	R	F1
True	0.81	0.86	0.83
Manipulated content	0.80	0.86	0.83
False connection	0.72	0.64	0.68
Satire	0.70	0.53	0.61
Misleading content	0.77	0.70	0.73
Imposter content	0.61	0.28	0.38
micro-average	0.76	0.73	0.74
macro-average	0.72	0.60	0.65
accuracy		0.78	

The BERT model outperformed all previous unimodal deep learning approaches, achieving 78% accuracy and 74% micro F1 score, demonstrating the advantage of pre-trained contextual text representations.
BERT performed best on the majority classes (True and Manipulated content, 83% F1), and worst on the minority class (Imposter content, 38% F1), mirroring the trend of previous models.
Surprisingly, BERT achieved better scores for Misleading content than Satire, despite Misleading being more represented in the dataset.
Overall, the strong performance of the BERT unimodal model highlights the benefits of using pre-trained contextual language models for misinformation detection.

Multimodal Approach:



Class	P	R	F1
True	0.85	0.88	0.86
Manipulated content	1	1	1
False connection	0.77	0.76	0.76
Satire	0.82	0.72	0.77
Misleading content	0.75	0.79	0.77
Imposter content	0.46	0.25	0.32
micro-average	0.88	0.86	0.87
macro-average	0.76	0.70	0.72
accuracy		0.87	

The multimodal approach, combining text and image inputs, achieved the highest accuracy of 87% and micro F1 of 72% compared to all the unimodal models.
The performance of the models was strongly influenced by the training set size for each class. The majority classes (True and Manipulated content) had the highest scores, while the minority class (Imposter content) had the lowest F1 of 32%, even lower than BERT's 38% for that class.
The image content provided little information for identifying Imposter content, but was very effective for Manipulated content, likely due to the detectable manipulations in the images.
The multimodal approach significantly improved the results for False Connection (76% F1, 8 points higher than BERT) and Satire (16 points higher than BERT and the unimodal CNN), demonstrating the value of combining text and image inputs.
Overall, the multimodal model outperformed the unimodal approaches, highlighting the importance of leveraging both textual and visual information for accurate misinformation detection.

Comparison of the best models:

Here in this documentation we have compared the performance of various models for misinformation detection, including traditional algorithms, unimodal deep learning models (CNN, BiLSTM, BERT), and a multimodal CNN approach which we have used for the fake news detection task.
The results show that the multimodal CNN model outperforms all the unimodal approaches, achieving an accuracy of 87% and micro F1 of 87%. This demonstrates the value of combining text and image inputs for accurate misinformation detection.
Among the unimodal models, the BERT model achieves the best performance, with an accuracy of 78% and micro F1 of 74%. This highlights the advantages of using pre-trained

contextual word embeddings from BERT, as opposed to the context-free GloVe vectors or random initialization used in the other deep learning models.
The BiLSTM model with dynamic GloVe vectors is the third-best unimodal approach, while the traditional SVM baseline performs the poorest among the models compared.
Overall, the results emphasize that deep learning approaches, especially the multimodal and BERT models, outperform traditional algorithms for the task of misinformation detection.

Conclusion:

Fake news can have significant negative effects, making rapid and reliable detection crucial. This is one of the few studies comprehensively comparing unimodal and multimodal deep learning approaches for fine-grained misinformation classification.
The key findings are:

- **The multimodal CNN model outperforms all unimodal approaches, highlighting the benefits of combining text and image inputs.**
- **Among unimodal models, BERT achieves the best performance, demonstrating the advantages of pre-trained contextual embeddings over static word vectors.**
- **Using dynamic GloVe embeddings improves over random initialization for CNN and BiLSTM architectures.**

As future work, we can do following things to improve the performance of the model:

- Explore pre-trained visual feature extractors like VGG.
- Experiment with various deep learning techniques like LSTM, GRU, and different fusion methods (late fusion).

References:

1) https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040
2) https://arxiv.org/pdf/2102.04458
3) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10006567/
4) Boididou, C., Andreadou, K., Papadopoulos, S., Dang-Nguyen, D.-T., Boato, G., Riegler, M., Kompatsiaris, Y. et al. (2015). Verifying Multimedia Use at MediaEval 2015. MediaEval, 3 , 7.
5) Breiman, L. (2001). Random Forests. Machine Learning, 45 , 5–32. doi:https: //doi.org/10.1023/A:1010933404324.
6) Brownlee, J. (). A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks. https://machinelearningmastery.com/ early-stopping-to-avoid-overtraining-neural-network-models/.
7) Deng, L., & Liu, Y. (2018). Deep learning in natural language processing. (1st ed.). Springer.
8) https://colab.research.google.com/drive/1edXYIghmzu3Bs4UhWJ8Ho9sk13oaebfz?usp=sharing

9) Baheti, P. (2020). Introduction to Multimodal Deep Learning. Retrieved from https://heartbeat.comet.ml/introduction-to-multimodal-deep-learning-630b259f9291. Accessed November 13, 2021
10) Brown, E. (2019). Online fake news is costing us $78 billion globall each year. ZDNet. https://www.zdnet.com/article/online-fake-news-costing-us-78-billion-globally-each-year/.