**Fake News Detection**

**Introduction:**

Fake news has become a potential threat to the socio-political fabric and has been resulting in chaos and public unrest. Fake news can be in wide variety of categories like in politics (show false infographics in order to bias a particular political entity), religion (to start a chaos between different communities), health (to spread false information to promote or demote certain products from particular companies) or to overthrow governments (by creating and spreading). Most of the fake news is spread through social media since it has the potential to reach wide audience across various communities and countries. We wish to grade any information that is passed through social media through Natural Language Processing Techniques by comparing the articles with authentic parallel sources (if corresponding sources are available). The topic being discussed in the articles will be extracted and are searched for relevant sources. After gathering relevant sources corresponding to the same topic, we would compare the texts and find out the dissimilarity between the articles. Based on this dissimilarity grading will be awarded to the articles. Based on the grade, users can decide for themselves if they are reading authentic content.

**Background:**

NLP is the area of artificial intelligence that focuses on teaching computers to comprehend spoken and written language in a manner similar to that of humans. NLP blends statistical, machine learning, and deep learning models with computational linguistics—rule-based modeling of human language. With the use of these technologies, computers are now able to process human language in the form of text or audio data and fully "understand" what is being said or written, including the speaker's or writer's intentions and sentiment.

Computer programs that translate text between languages, reply to spoken commands, and quickly summarize vast amounts of text—even in real time—are all powered by NLP. You've probably used NLP in the form of voice-activated GPS devices, digital assistants, speech-to-text dictation programs, customer service chatbots, and other consumer conveniences. The use of NLP in corporate applications, however, is expanding as a means of streamlining business operations, boosting staff productivity, and streamlining mission-critical business procedures.

It is extremely challenging to create software that reliably ascertains the intended meaning of text or voice data since human language is rife with ambiguity. Homonyms, homophones, sarcasm, idioms, metaphors, exceptions to the rules of grammar and usage, and changes in sentence structure are just a few examples of the irregularities in human language that take humans years to learn but that programmers must teach natural language-driven applications to recognize and understand accurately from the beginning if those applications are to be useful.

**Speech Recognition-**

The process of accurately translating voice input into text is known as speech recognition, commonly referred to as speech-to-text. Any program that responds to voice commands or questions must use speech recognition. The way individuals speak—quickly, slurring words together, with varied emphasis and intonation, in various dialects, and frequently using improper grammar—makes speech recognition particularly difficult.

**Part of speech tagging-**

The act of identifying a word's part of speech based on its use and context is known as part of speech tagging, also known as grammatical tagging. In the sentences "I can create a paper plane" and "What make of car do you own?," the word "make" is classified as a verb and a noun, respectively.

**Word sense disambiguation-**

It is the act of choosing a word's meaning from among its possible meanings using semantic analysis to discover which word makes the most sense in the context at hand. Word sense disambiguation, for instance, clarifies the difference between the meanings of the verbs "make" and "make the grade" (achieve) and "make a bet" (place).

**Named entity recognition-**

Words or phrases are recognized as useful entities using named entity recognition, or NEM. NEM identifies "Kentucky" as a place or "Fred" as the name of a guy.

**Co-reference resolution-**

The task of determining whether and when two words refer to the same item is known as co-reference resolution. The most typical example is figuring out who or what a certain pronoun refers to, but it can also require figuring out a metaphor or idiom that is used in the text (e.g., when "bear" refers to a big, hairy person rather than an animal).

**Sentiment analysis-**

It looks for intangible elements in text, such as attitudes, feelings, sarcasm, confusion, and suspicion.

**Natural language generation-**

It is the process of converting structured data into human language; it is frequently referred to as the opposite of voice recognition or speech-to-text.

[]https://www.ibm.com/cloud/learn/natural-language-processing#:~:text=Natural%20language%20processing%20(NLP)%20refers,same%20way%20human%20beings%20can

**Top 3 models used in NLP-**

**BERT (Bidirectional Encoder Representations from Transformers)-**

BERT developed by Google, uses the Transformer, a cutting-edge neural network design for language interpretation that is built on a self-attention mechanism. It was created to deal with the issue of neural machine translation or sequence transduction. In other words, it works best for any activity that converts an input sequence into an output sequence, like text-to-voice conversion or speech recognition, for example.

The BERT algorithm has successfully completed 11 NLP tasks. It was trained using 800 million words from the BookCorpus dataset and 2,500 million words from Wikipedia. One of the best illustrations of BERT's effectiveness is Google Search. BERT is used for text prediction in other Google products, like Google Docs and Gmail Smart Compose.

**RoBERTa (Robustly Optimized BERT Pretraining Approach)-**

The language model in RoBERTa is based on BERT's language masking technique, which enables the system to learn and anticipate purposely disguised text segments.

By training with larger mini-batches, deleting BERT's next phrase pretraining target, and other changes, RoBERTa alters the hyperparameters in BERT. Pre-trained models like RoBERTa, which may be used for NLP training tasks including question answering, dialogue systems, document categorization, etc., are known to outperform BERT in all individual tasks on the General Language Understanding Evaluation (GLUE) benchmark.

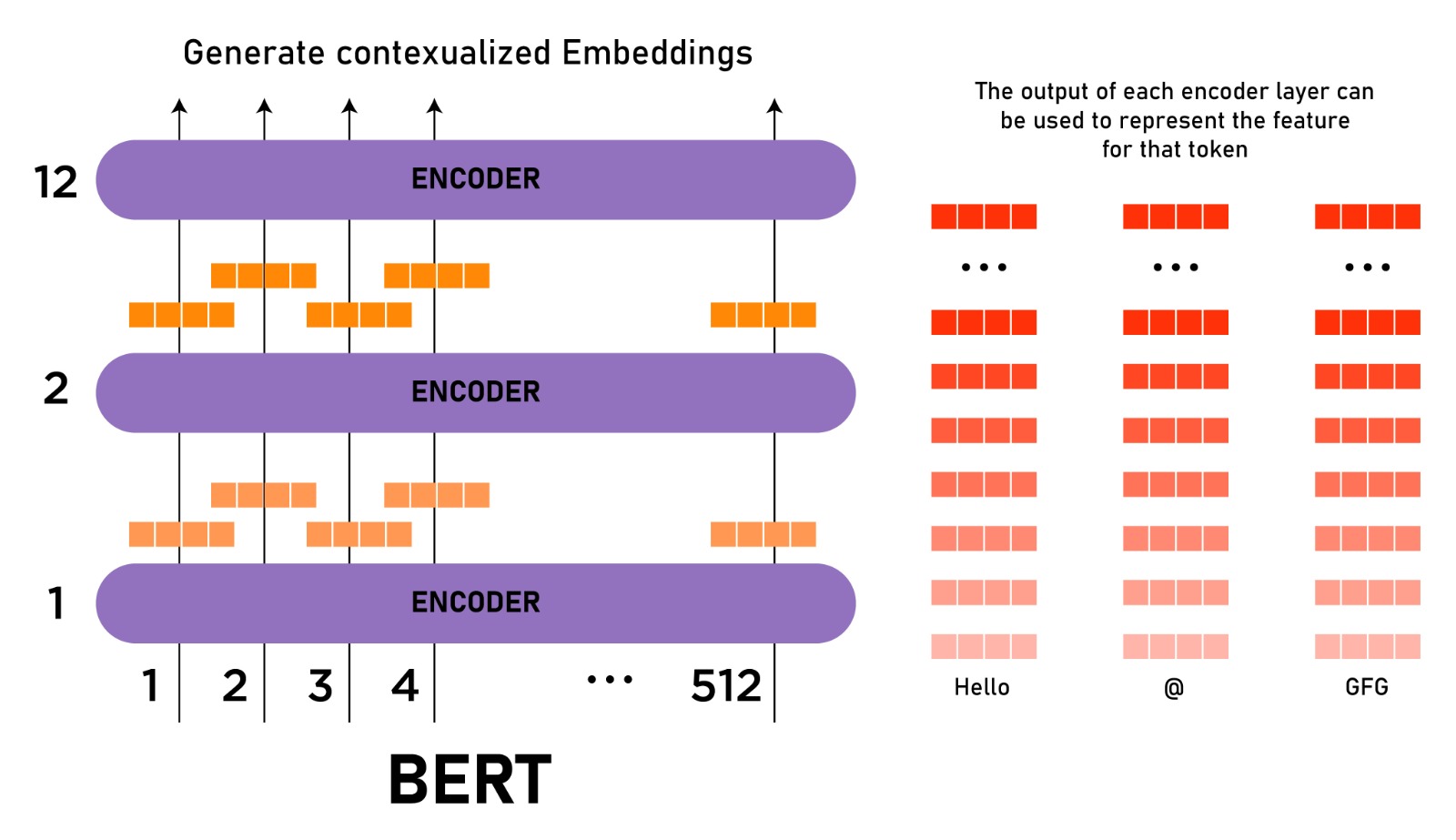
**OpenAI’s GPT-3-**

The GPT-3 model uses transformers to accomplish tasks like translation, question-answering, poem creation, cloze, as well as those that call for quick decision-making, including word unscrambling. The GPT-3 is also used to develop codes and write news stories thanks to recent advancements. GPT-3 can control statistical relationships between various words. It is trained using 45 TB of text collected from around the internet and over 175 billion parameters. This makes it one of the most comprehensive pre-trained NLP models accessible.

[5]

**Our Model:**

**Architecture Diagram**

****

[6]

Bidirectional Encoder Representations from Transformers, or BERT, is a deep learning model that is based on Transformers. In Transformers, each output element is connected to each input element, and the weightings between them are dynamically determined based upon their connection.

NLP technique aims to comprehend spoken human language in its natural setting. For BERT, this often entails picking a word out of a blank. Models must typically be trained using a sizable collection of specific, labeled training data to accomplish this. This calls for teams of linguists to laboriously label data by hand.

However, BERT was only trained using on unlabeled, plain text corpus. Even while it is being utilized in practical applications, it still learns from the unlabeled text and continues to advance. Its pre-training acts as a foundational layer of "knowledge" upon which it's built. From there, BERT can be adjusted to the user's preferences and then constantly expanding its body of searchable content, which essentially is Transfer learning.

The transformer is the component of the model that gives BERT its improved ability to comprehend linguistic ambiguity and context. Instead of processing each word individually, the transformer accomplishes this by analyzing each word in relation to every other word in the sentence. The Transformer enables the BERT model to comprehend the word's complete context and, as a result, better understand the searcher's intent by taking a look at all the surrounding terms.

This contrasts with the conventional approach to language processing, called word embedding, in which earlier models like GloVe and word2vec would map every word to a vector, which only captures a little portion of its meaning in one dimension.

Large amounts of labeled data are needed for these word embedding models. However, because all words are in some way tied to a vector or meaning, they struggle with the context-heavy, predictive nature of question answering. To prevent the word in focus from "seeing itself," or having a fixed meaning independent of its context, BERT employs a technique of masked language modeling. The masked word must then be determined by BERT only based on context. When using BERT, words are defined by their context rather than by a predetermined identity.

The bidirectional Transformers allow it to be the first NLP technique to entirely rely on self-attention mechanisms. This is important since a word's meaning frequently changes as a phrase progresses. The total meaning of the term that the NLP algorithm is focusing on is enhanced by each additional word. The word in focus becomes more uncertain the more words there are overall in a sentence or phrase. By reading in both directions, taking into consideration how all other words in a sentence affect the focus word, and removing the left-to-right momentum that causes words to be skewed towards a particular meaning as a sentence advances, BERT compensates for the augmented meaning.

[7]

Diagram

Description automatically generated

Workflow Diagram description

**Dataset:**

Our data are json files stored on the web i.e google api public resources therefore we download it and convert it into pandas dataframes. It has news about a plethora of topics. Here we have 800 rows and 3 columns for the training dataset and we use only text column for our modeling. The main feature of using google api resources is that it has news from different websites and the URL, title and content is clearly viewed.

Text

Description automatically generated with medium confidence

Same as the training set , for the test set we have 800 rows and 3 columns.

Word

Description automatically generated with medium confidence

One of the benefit of using google api resources dataset is that in case if model doesn't perform well we can use another feature/column for example URL for better modeling.The number of real news and fake news in our dataset are same so,

it won't be an imbalance classification problem.

**Analysis of data:**

**Data Pre-processing:**

The dataset we considered has a balanced amount of data of real and fake news. It has 3 columns initially, namely URL, title, text. We label our data where real news is labelled as 0 (negative) and fake news are labelled as 1 (positive). The reason we label fake news as positive is that the main purpose of the modelling is to detect fake news.

A screen shot of a computer

Description automatically generated with low confidence

Except for the text column, we remove the other columns in this case. We then remove non alphanumeric characters as well as converting to all lower case from the text. We use train set to perform exploratory analysis. First, we want to look at the word count for each news and see if there is difference between real and fake news.

Chart, histogram

Description automatically generated

We can see from the above graph that most real news is within 1000 words, and the distribution of word count is skewed to the right. As for the fake news, we see some outliers from above figure, making it hard to interpret, so we plot it again below with outlier (news that has more than 20,000 words) removed. Next, we like to see what the most common words in real/fake news are to discover some patterns. Word cloud is a popular way to visualize it.

Text

Description automatically generated

We can see most of the real news are about COVID19 virus, and the common words are countries name and some neutral words.

Text

Description automatically generated

As for fake news, the topic is also the same. However, it contains some strong words such as biological weapon, as well as some names such as Donald Trump and Bill Gates. Next, we would like to see the took word proportion of the real/fake news. In other words, we like to see how many of the words used in the news are from the top 10 common words, top 100, and so on. The reason to do so is that we suppose fake news are machine generated and it use many high frequency words comparing to real news. For this purpose we have used CountVectorizer sklearn.feature\_extraction.text package. This is used to convert a collection of text document to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr\_matrix. If you do not provide an a-priori dictionary and you do not use an analyzer that does some kind of feature selection then the number of features will be equal to the vocabulary size found by analyzing the data.

Chart, bar chart

Description automatically generated

Fake news is slightly more likely to have top frequent words, but the difference is not significant. Since, BERT algorithm can only accept sentence length up to 512 words, we need to pre-process our data (long news) in order to feed in to the algorithm.

Next, we pre-process our original text into input features BERT can read. The words are tokenized base on the vocabulary dictionary it pretrained on (about 30,000 words). Unknown words are broken down into smaller words contained in the dictionary. Maximum sequence length is also specified so we can pad all sequence into the same length.

**Implementation:**

For the process of detection of fake news we segregate the data using o’s and 1’s which is real(negative) and fake(positive) respectively. We now check the word count of the news to differentiate between what is real and what is not. Real news mostly consists of less amount of words. A set of news with an abnormal amount of words is often removed. By this we can filter out most of the context of the news. Now, we can check for the most common words so that we can determine a pattern in the news. We can use word cloud as the medium to conceptualize the data. We can consider a topic and search for the words which are more relevant to that topic. The fake news will also have the same topic but it is going to incorporate words which are too sophisticated. To eliminate this problem we can consider the most common words. Fake news is mostly machine generated and it uses words repeatedly compared to the real news. Packages such as sklearn.feature\_extraction.text are used form the CountVeactorizer library which is used to form a matrix which consists of the token counts. Using BERT model we can limit the amount of words used in the news context. In order to implement this we separate the text in to subtexts with a limited amount of words. We now tokenize words and create token ids for BERT algorithm. The BERT model will filter out all the unknown words with which the original sequence would have been longer but when we breakdown the words which are unknown we can split words into 2 words which act as prefix and suffix. The token which do no tend to appear in the original word count the get assigned as the UNK which is a special token. After the segregation of the words all the token are provided with token IDs. These act as the input to the BERT transformer in which there is a mask for every sequence. The mask contains approximately about 30% of the redundant words. The BERT model only takes 512 word embedding. The encoder then uses the input from the classified text and produces vectors which are eventually multiplied using the embedding matrix to obtain the original vocabulary dimension. Here the softmax function is used to perform a probability calculation in the obtained vocabulary matrix. The dropout layer sees that any overfitting is avoided by considering a few nodes and disabling them. This way we can achieve the best precision, recall and f1 scores, possible.

**Results:**

A screenshot of a computer

Description automatically generated with low confidence

The BERT model was tuned to get higher recall than precision. More recall meaning, it is less likely to get false negatives. In this type of problem, it is mandatory to capture maximum number of positives even though it poses a risk of attracting False positives and reducing the precision. So the model is more likely to predict a real news as fake than predicting fake as real news. This way users can revalidate the authenticity of the news rather than simply believing it.

**Project Management:**

Pranay – Collected the dataset and research about the algorithm

Sripathi – Analyzed the data and plotted graphs

Aditya – Designed the algorithm as per the requirement

Aneesh – Tested analyzed and made observations for the model.

**References:**

1. <https://www.analyticsvidhya.com/blog/2019/12/detect-fight-neural-fake-news-nlp/>

2.<https://medium.com/analytics-vidhya/fake-news-detection-using-nlp-techniques-c2dc4be05f99>

3.<https://www.researchgate.net/publication/335191041_A_Machine_Learning_Approach_to_Fake_News_Detection_Using_Knowledge_Verification_and_Natural_Language_Processing>  
4. <https://arxiv.org/pdf/1901.09657>

5. <https://insights.daffodilsw.com/blog/top-5-nlp-language-models>

6. <https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/>

7.<https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model#:~:text=BERT%2C%20which%20stands%20for%20Bidirectional,calculated%20based%20upon%20their%20connection>

GITHUB Link

<https://github.com/PranayNagavolu/fake-news-detection>