Fighting Health-Related Misinformation in Social Media With Large Language Models

By

Moisés Robles Pagán

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2024

Approved by:

Manuel Rodríguez Martínez, Ph.D. President, Graduate Committee	Date
Emmanuel Arzuaga, Ph.D. Member, Graduate Committee	Date
Domingo Rodríguez Rodríguez, Ph.D. Member, Graduate Committee	Date
FirstName I. LastName, Ph.D. Representative of Graduate Studies	Date
José Cedeño Maldonado, Ph.D. Department Chairperson	Date

ABSTRACT

Combating disinformation in social media is a critical problem, notably when the disinformation targets healthcare. We explore how to fine-tune Large Language Models (LLM) to counteract health-related disinformation on social media. The fine-tuned base models for this project are T5, BERT, and LlaMa-2. We divide the fine-tuning into two sections: 1) classifying if the text is health-related and 2) verifying if the text contains disinformation. To rebut disinformation we use Retrieval Augmented Generation (RAG) to query trusted medical sources. Our experiment shows that the models can classify health-related with 94% precision, 95% recall, and 90% F1. We also show that we classify disinformation texts with 99% precision, 95% recall, and 97% F1. We present a project that can help health experts combat and rebut disinformation on different social media platforms.

RESUMEN

El Resumen debe ser una traduccion del Abstract. No deben diferir en contenido.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

ACKNOWLEDGMENTS

I want to thank the GRIC personnel! :D

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Contents

List of Figures

List of Tables

List of Abbreviation

AI	Artificial Intelligence
API	Application Program Interface
CLM	Causal Language Modeling
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DBMS	Database Management System
ETL	Extract, Transform, and Load
GPU	Graphic Processing Unit
HF	Hugging Face
JSON	JavaScript Object Notation
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short-Term Memory
MLM	Mask Language Modeling
NIH	National Institutes of Health
NLP	Natural Language Processing
PEFT	Parameter Efficient Fine-Tuning
PMC	PubMed Central
RNN	Recurrent Neural Network
THS	Twitter Health Surveillance

UPRM University of Puerto Rico Mayagüez

Chapter 1

Introduction

1.1 Motivation

Nowadays, technology has advanced to the point that anyone can find any information in just a few seconds. Social media has been an essential element in the search for information. The issue with this is that anyone can find, search, share, and even write anything, accurate or not. However, this dilemma has caused problems in this modern era. If anyone can share anything, how can you be sure what is true? Users are susceptible to disinformation or misinformation. In this context, misinformation refers to messages with false information dispersed because the author misunderstood facts. In contrast, disinformation refers to messages with false information that are intentionally dispersed. The author of these messages has the intention of forming opinions based on false data. In either case, false information spreads to readers as facts. Most social media platforms recommend that readers read from experts or official news outlets. Nevertheless, the overwhelming amount of data makes it complicated to keep up with everything.

Currently, social media such as X (formerly known as Twitter) have ”Community

Notes" which clarify tweets that are misleading or misinforming. However, this system depends totally on human interaction and is a slow and intricate process. On most occasions, when a "Community Note" is added to a tweet, the disinformation has already been spread. The issue of detecting and preventing the spread of misinformation has not been an easy task, especially in the health field. Another important factor mentioned in [1] is that health-related textual misinformation is more engaging when compared with pictorial.

In recent times, it has been challenging for health officials to achieve the prevention of endemic or pandemics. Most of the time, these officials tend to make educational campaigns for the population. However, social media misinformation can reduce the effectiveness of these campaigns. In addition, this is harder to counteract because these can spread for longer times and reach different users [1]. Some problems these experts have faced in the past years were misinformation about vaccines, users invalidating safe measurements, and other issues.

The Twitter Health Surveillance (THS) system was designed to detect tweets related to health conditions [2]. THS is a prototype system we are building at the University of Puerto Rico, Mayagüez (UPRM). The project is designed as an integrated platform to help health officials collect tweets, determine if they are related to a medical condition, extract metadata from them, and create a warehouse that can be used to analyze the data further. The THS Artificial Intelligence (AI) components used Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) to classify tweets as being medically related, unrelated, or ambiguous. The data they used was from the Twitter API system, then processed through the Hadoop ecosystem and stored in a Hive database.

Searching for training data about this topic is not an easy task. A problem with social media text is the informality, slang terms, or special characters. However, we use the data from the THS project as our primary source for the health classification

dataset. On the other hand, the misinformation dataset contains social media posts, articles, websites, and others. Instead of using the THS architectures, we opted for Large Language Models (LLM).

An LLM is a computational model with high capability in Natural Language Processing (NLP) thanks to its ability to make statistical relationships within texts. These models have made breakthroughs in how computers interpret human language. Additionally, we can train them to accomplish text-based tasks. Some examples are classifications, answering questions, making inferences, summarizing, and others. Thus, we can train them to classify and make inferences from a text.

We employ a transformer-based LLM in this project. These are effective in the natural language process field (NLP) [3]. The system detects health misinformation and provides context for its classification. Also, we did not preprocess the tweets because they could lose the context of the actual meaning if we remove hashtags, mentions, and emojis. This system has a high chance of detecting misinformation and rebutting it to educate users. We built the prototype using Python, PyTorch, Chroma, Ollama, and other open-source tools. In determining if a text is health-related, our system achieved a 90% F1 score and a 97% F1 if it contained misinformation. Additionally, our preliminary results show that using official health sources with Retrieval Augmented Generation (RAG) helps the LLM rebut correctly. Hence, our system proved that it is possible to classify and rebut health-related misinformation.

1.2 Objectives

The objectives of this project are as follows:

- Identify and extract information from official health sources: This data will be stored in a vector database that the model will use as context to rebut the mis-

information. The model will cite official health sources related to the tweets to sustain their classification.

- Identify and finetune a Large Language Model: Select an appropriate base Large Language Model architecture that will:

  1. Detect if a text is health-related.
  2. Determine if a text is misinformation.
  3. Use official health sources texts to combat the texts classified as misinformation and cite from the gathered data.

- Compare with the previous version of THS: To measure the effectiveness of the classification with the LLM, we are going to compare it with the previous THS results and validate the advantages of a Large Language Model in solving Natural Language Processing problems.

1.3 Contributions

- **Leveraging LLMs for Health Misinformation:** Large Language Models are being used for different fields nowadays. However, these do not focus on health misinformation on social media. We present Large Language Models as a solution to classify and rebut health misinformation texts on social media and use research papers extracted from PubMed as context for the LLM.

- **Health and Misinformation Classification Datasets:** We used 12,441 texts for the health-related classification labeled as related, unrelated, or ambiguous. For the misinformation-classification we had 8,772 texts labeled as misinformation or not. For the model rebuttal, we extracted 56,365 papers from PubMed.

- **Present a novel solution to misinformation rebuttal:** For misinformation rebuttal, is necessary to have an understanding of what needs to be fact-checked. Also, it is important to have the necessary context for the correction. We extracted research papers that were added to a vector database. The database helped find similar chunks of texts to use as context for the misinformation rebuttal. That setup enable us to use RAG to answer health misinformation with peer-review documents.

- **Pipeline Interface:** Developed a frontend application that showcase the full pipeline, allowing users to view the process.

1.4 Outline

This paper has the following organization. Chapter 2 contains the literature review on Transformers, Large Language Models, the different use cases of these models for classification, misinformation on social media, and the THS project. Additionally, we describe the importance of disinformation on social media. For Chapter 3, we can observe the problem description and methodology. Later, in Chapter 4, we have our system architecture and the project pipeline for the training and classification. Chapter 5 presents our results based on accuracy and performance. In Chapter 6, related works are presented, with our conclusion and suggestions for future work.

Chapter 2

Literature Review

In this chapter we provide the technical background related for the context of the project.

2.1 Large Language Models (LLM)

Natural language processing (NLP) has always been an intricate field because of the complexity of how humans communicate. The meaning of a message can vary because of homonyms, tone, context, and other factors that affect the message delivered. These are some challenges that computers face when trying to replicate or learn human text communication and expressions. However, this changed with the introduction of Large Language Models (LLM) [4]. These models are trained with large amounts of data to replicate human-like patterns or generate text based on statistical relationships between words, and these advancements were made possible by transformers [5]. Previous NLP techniques such as Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM) could help in understanding a sentence's context in the short term [6]. However, these struggle when trying to understand longer texts. In contrast, transformer architecture, seen in Figure 2.1, differs from others because it uses self-attention.

2.1.1 Attention Mechanism

This self-attention finds dependencies between all words in a text, short and long-term. The process of this starts by turning words into tokens. A token can be a word, subword, individual letter, or a sequence of words mapped to an embedding. The embedding is the vector representation of a token in high-dimension space; its size depends on how much information it stores about the token. Now, self-attention finds relation-

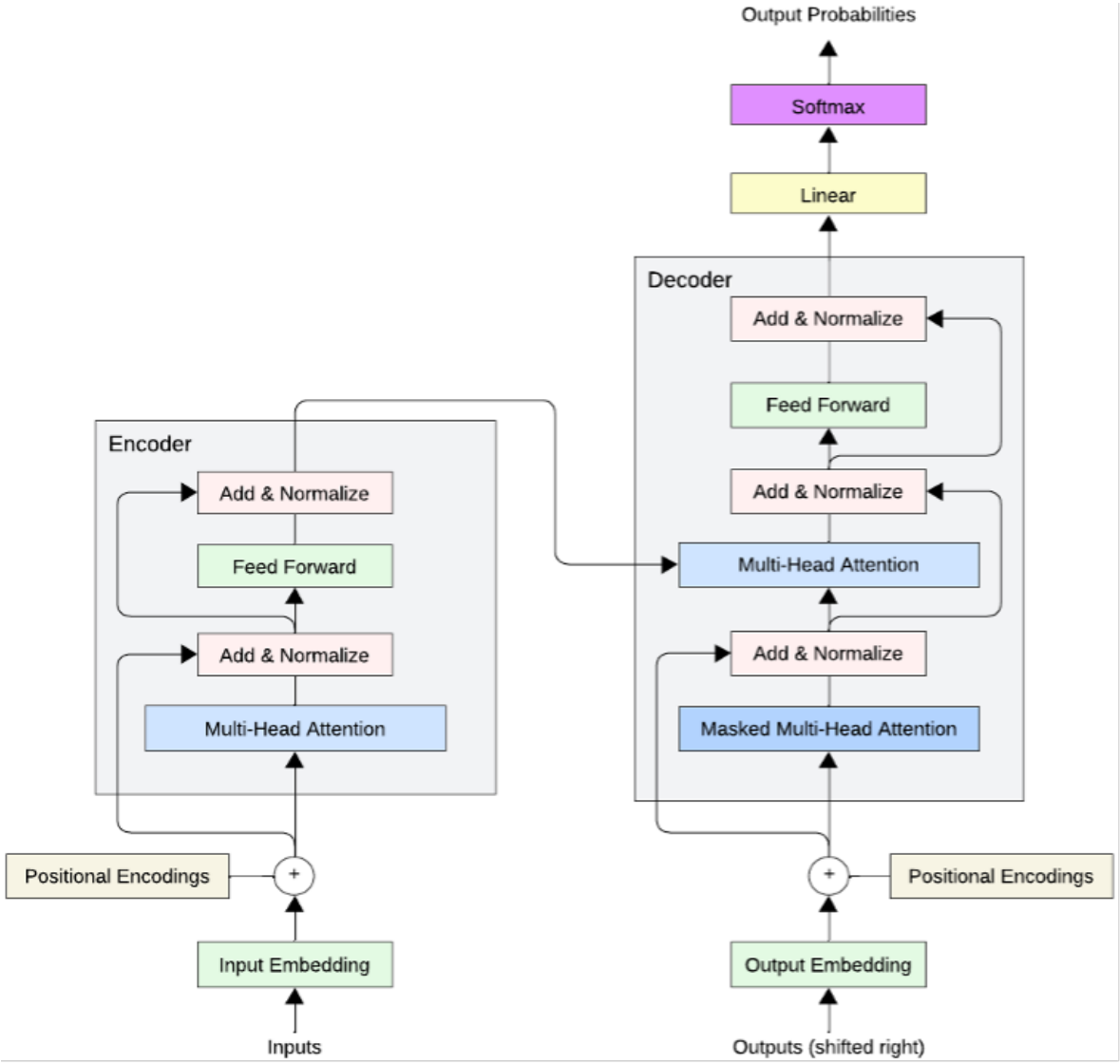

Figure 2.1: The Transformer Architecture

ships between all tokens and gives them an attention score, measuring the relevance of a token to others. Transformers uses the scores to generate a final representation of each token. This process depends on how the models make a token. The tokenization strategy is determined by the preprocessing stage, and influenced by the embedding and model architecture. The embedding impacts the strategy because of its dimensionality, the amount of information encoded, and the model's sequence length limitations.

2.1.2 LLM Architecture

In Figure 2.1, we can see an encoder and a decoder in the transformer architecture. Based on that, there are different LLM architectures: decoder-only, encoder-only, and encoder-decoder. Each one has advantages for specific tasks and limitations for others.

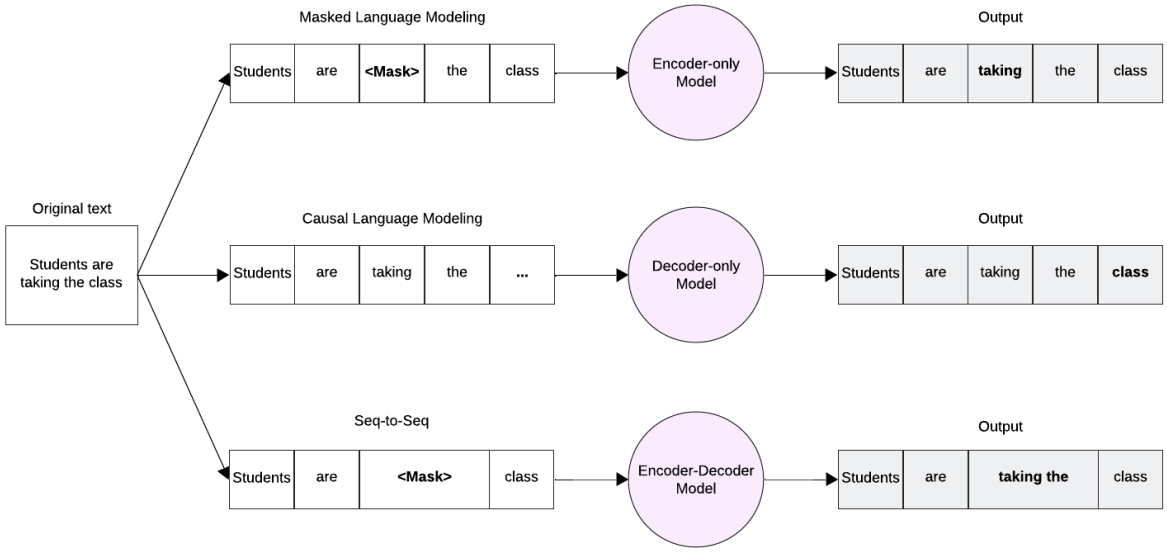

Figure 2.2: LLM Architecture and Comparison

Decoder-only models: Decoder-only models predict the next word based on the previous context, thus being unidirectional models. The model achieves this by taking a text or prompt as input and returning a first word. For each subsequent word, the model uses the previously generated text as input to predict the next word,

continuing until it produces a coherent output. In Figure 2.2, the model predicts the last word based on the previous context. Because of their ability to predict sequences of texts, they are frequently employed in tasks like summarization and text generation. Models that perform those tasks are called causal language modeling (CLM) because they predict new tokens not found in the input. Compared to the other architectures, these models are massive in size. Because of their sizes, these are not very practical or cost-effective for daily usage. These are the most commonly known models, such as GPT-3 [7], Mistral [8], and LLaMa [9].

Encoder-only models: These models predict by masking specific words in a sentence. Said masking helps them understand the meaning or relation of the masked word based on context. They are bi-directional, which means they take the context before and after the masking to evaluate the word. The example in Figure 2.2 shows a sentence with a word mask being inputted to the encoder LLM; the model predicts the missing word by using the surrounding text. That is why they tend to perform well at classification and sentiment analysis but are not optimal for text generation. Said models are called masked language models (MLM). In contrast to other architectures, these models are relatively small. Some examples of encoder-only LLM are BERT [10] and RoBERTa [11].

Encoder-Decoder models: These models combine masking and text generation. They work by masking the sequence of texts and using the context around it to make a prediction. As seen in Figure 2.2, they can mask more than one word from the original inputted text. Because the model generates sequences of texts, it is commonly used for translation and question and answer. Translating between languages cannot be achieved through word-for-word conversion; it requires understanding the entire sequence to preserve the context. To have an optimal model, both encoder

and decoder must be trained for the task one wishes to achieve. Depending on the task, it can be harder to train compared to the other types of architectures. Bart [12] and T5 [13] are examples of encoder-decoder LLMs.

Knowing the architectures, each one has its advantages for specific NLP tasks. However, to perform these tasks, we must train them to do so.

2.1.3 Classification tasks

Large Language Models use different learning methods to train. Such methods include zero-shot, one-shot, and few-shot learning. As the name implies, zero-shot learning is training a model without previous knowledge of the data; it is learning from scratch. One-shot learning is a method that receives one example as input and tries to generalize from that example. The final method uses multiple examples to find a pattern between them.

In Figure 2.3, we can see an example of zero-shot learning. This LLM has no previous knowledge of the task that it must perform. Nonetheless, the model used, GPT-3, has the advantage that it can follow instructions when redacted clearly. Here, we tell the model that it must act as a medical expert and identify if a message is health-related and why it is classified that way. Also, the result must follow a specific format. This process of instructions is called prompt engineering. Prompting does not retrain or adjust the model parameters. Thus, it does not always have optimal results.

Moreover, LLMs that completed training with no additional modifications are known as based models. The response from this model will most likely make no sense with the premises it receives. This happens because the model is trained on excessive texts to find patterns between them, but this pattern might not be on par with the input. For the model to return a coherent response, it must go through a fine-tuning process.

```
In [4]: prompt = f"""
        You are a medical expert and trying to identify if this message
        is health related. You will use the following format:

        Id: (number)
        Text: (write the message)
        Classification: (Related, Not Related, or Ambigous)
        Reasoning: (Why you classified it that way)

        Message: '''Got the flu :| tired face :|'''
        """

        response = get_completion(prompt)
        print(response)

Id: 001
Text: Got the flu :| tired face :|
Classification: Related
Reasoning: The message mentions having the flu, which is a health-
related issue. Additionally, feeling tired can be a symptom of the
flu.
```

Figure 2.3: Zero-shot example of GPT-3

Fine-tuning consists of making a model perform specific tasks, such as chatting, summarization, chatbots, and others. To fine-tune a model, it must undergo another training process, with training data related to the tasks it will perform. However, there are multiple training types. We will focus on Sequence Classification and Causal Language Modeling.

- **Sequence Classification:** This involves predicting a class label based on an input. That label is a number associated with a class. All previously mentioned models can be trained for Sequence Classification, which is the most common training type.

- **Causal Language Modeling:** This training is used for the model to learn the context of the text. We use this so the model learns to generate a text answer based on the input. The labels are texts related to the input. This type of training is similar to question-and-answer problems, in which the model infers a result based on the question. CLM is more expensive in computing power than Sequence

Classification because the labels must be embedded. That training type is most commonly used in models that have decoders because they are usually used to generate texts.

A problem with fine-tuning LLMs is that they require excessive computational resources to train or fine-tune. It is not viable to fine-tune all the parameters in the model to perform a specific task. However, there is an alternative to fine-tuning a model without having a resource problem.

2.1.4 Low-Rank Adaptation (LoRA)

The research in [14] created a technique to train models with billions of parameters effectively. Their technique, LoRA, helps reduce the amount of computational resources and parameters to train while maintaining optimal performance. LoRA trains specific layers of the model instead of retraining the full system. Their research shows that they reduced the GPT-3 training size from 350GB to 35MB. However, a limitation of LoRA is that we must add the trained layer to the base model. That means that after training, we use both the base model and this adapted layer to perform a task. Nonetheless, it is still an improvement when they reduced the resources used by 3x, and their model had an accuracy of $\pm 0.5\%$ compared to full fine-tuning. An effective fine-tuning process is possible, but we need to see some use cases for these models.

2.1.5 Example Applications

The authors in [15] trained a based LLM model to understand images and give a text description or a combination of text and image. The training process for the model used images and their captions as their data and the zero-shot learning strategy. Their resulting model was used as a chatbot that identifies images, answers questions, or gives

visual examples. Another experiment was [16], where the authors trained a model to identify sentiments on financial market decisions. In this case, they used prompting, also called in-context learning, for the model to answer the sentiment of the texts. In-context learning does not update any parameter of the original model. Their model resulted in a 70% accuracy on sentiment prediction, failing mostly on neutral posts. A possible problem with the neutral post is that prompting does not train the model to perform a specific task. That experiment showed the problem that users face when they use social media to make decisions. They clarified the importance of not taking the model for granted and how social media can cause a user to make a poor decision.

2.1.6 Challenges and Limitations

An issue with LLM is that they answer based on statistical relationships between words, and occasionally, their output could make no sense. Sometimes, they can generate a result that is not factual or valid; this phenomenon is called AI hallucinations. These models train to find patterns between tokens. Without the proper context, they cannot differentiate between fact and fallacy. This context needs to be related or similar to the input that the model receives. However, finding the necessary information is not optimal if done manually. That begs the question, how much data would be needed to achieve a coherent answer, and how can this be done effectively?

2.2 Vector Databases

AI hallucinations occur when models generate a plausible output but lack factual accuracy. We can minimize the hallucinations by providing context relating to the text inputted. A solution to this is finding data from officials or experts on the topic associated with the text. Given the vast amount of research available, narrowing down

relevant information becomes a challenge. In addition, the context quality is essential for the model to make any inference. The search process for this consumes time and is not always optimal.

Nonetheless, if we have an immense amount of data from different sources, we can benefit by using a vector database. In contrast to other databases, they store data in a vector representation. These databases can come in different forms, like Chroma [17], a native vector database. Also, some relational databases have extensions that allow them to make vector similarity searches, like Postgres with pgvector [18]. They can turn images, videos, documents, and others into numerical embedding. These embeddings are numerical vectors capturing semantic meaning [19]. They are practical because the system stores the data provided as vectors in a high-dimensional space, where semantically similar data are positioned close in said space. We can see a text example of the vector space in 2.4. If we search for "Flu" it could return "Sick" or "Covid" with a higher probability than "Car" or "Truck". As we can see, querying this database will return a result close to the input.

Although these systems help find similar text, there are a few limitations. One such problem is the vector size, which can impact the accuracy and resource usage [20]. If the vector size is relatively big, the system will have a higher accuracy but will require more storage and resources to process. Also, this affects when searching large datasets, which incurs a resource-intensive search.

For the data insertion process, we need to find optimal strategies for our data. If we focus on a text dataset, the best practice is to split the document into chunks. This splitting process must be optimal for the task we want to achieve. If we split it into small chunks, it will have a more precise answer, but we dilute the meaning of the text. On the other hand, larger chunks can have more context but less relevance. After selecting the appropriate size, these chunks must be turned into a vector, usually done

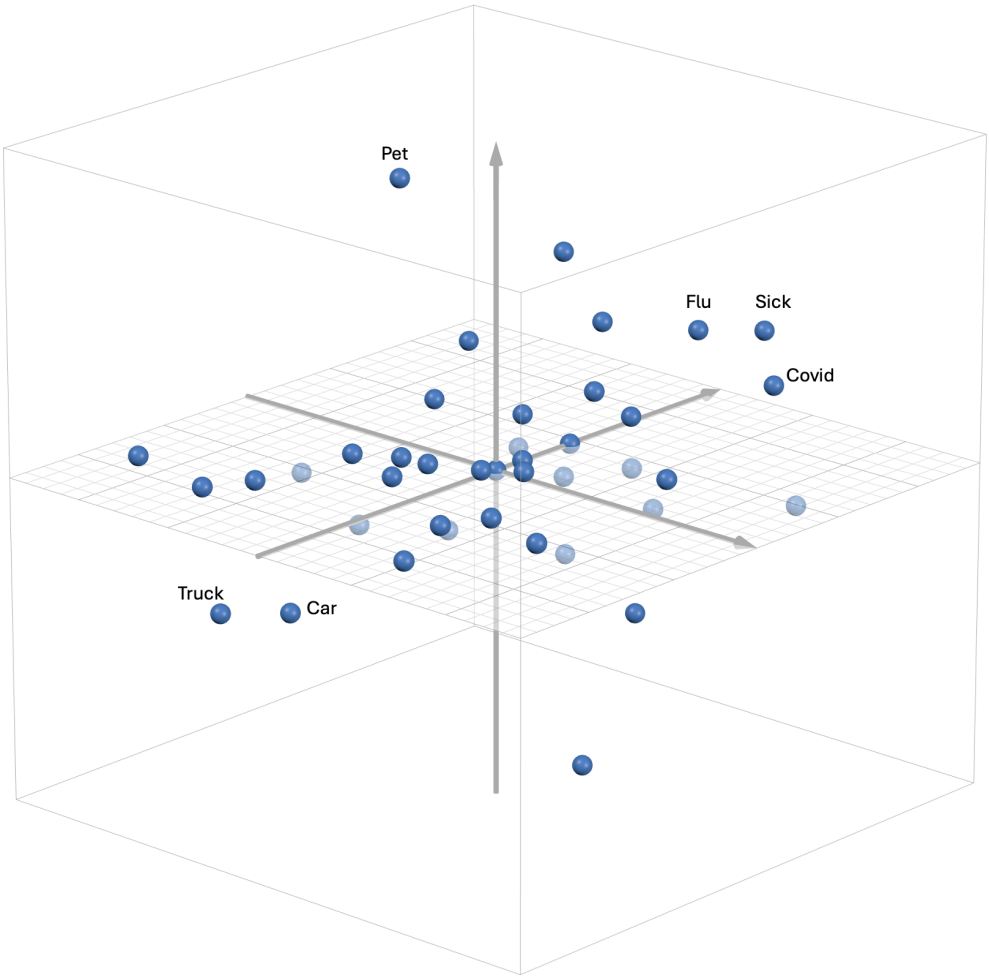

Figure 2.4: High-Dimensional Space Representation

with an LLM. Finally, we store these vectors and their chunks in the database system.

To search the system, we send a query to the LLM that converts it into a vector. Then, the vector is used in the database to make a similarity search. The system will return chunks that can function as context for an LLM to analyze. In this context, an LLM can use the retrieved chunks to generate a fact-based answer from outside their initial training. This process, known as Retrieval-Augmented Generation (RAG), enables LLMs to access relevant information during text generation. In [21] they tested LLM to answer a test that contained images and text. They evaluate the GPT3 base model, the base model with prompt, and a GPT3 with RAG. Their experiments showed

that the base model's average success ratio was 40%, the second model was 58%, and the model with RAG ended with 75%. Said experiment showed that an LLM can outperform when it receives the necessary context. Therefore, RAG can assist in providing expert-level responses, but human oversight is still imperative for complex topics. Additionally, in a world with many sources of information that can be misleading to people, this can help identify texts that are not factual or incorrect.

2.3 Misinformation in Social Media

There are many sources in the world to find information about any topic. Nonetheless, many people use social media as their primary source [22] and occasionally take this information as truth without validation [23]. On occasion, these can be fake, misleading, or wrong. When this happens unintentionally or by lack of understanding of the topic, it is called misinformation. On the other hand, when it is intentional to provide wrong information, this is known as disinformation. For simplification, both terms will be used interchangeably, as they have a similar impact on the user and give information that is not accurate.

Misinformation has been dangerous during critical events like natural disasters or health crises. For instance, during the COVID-19 pandemic, false claims appeared saying that the vaccine had microchips or that it was intended for population control [24], which led to high health risks or even deaths [1] because they refused to get vaccinated out of fear. A problem with disinformation is that the audience does not always detect it. When misinformation spreads and is not clarified early on, it can be confused as fact. Misinformation can affect all demographics, but older audiences and people with less education are more likely to share and believe fake or misleading news [25].

There have been various research studies on reducing the propagation of misinfor-

mation. Misinformation was modeled as a game-theoretic problem in [26], where some players spread fake news, and others tried to stop it. They created an agent at the network level to combat misinformation in a simulation. However, they could not conclude the efficiency of their model due to the lack of discernible patterns in the simulation. On the other hand, the authors in [27] used LSTM and BERT to classify misinformation from the different news sources. They proved that BERT outperformed LSTM, achieving an accuracy of 64.88% against 60.59%. These results are significant in detecting misinformation; still, they are not optimal for situations that can directly impact someone's life. For example, the system has over a third chance of misclassifying news, and this could be dangerous if the topic is a natural disaster or health risk. We need to ensure that AI models classify this type of news with a very high accuracy rate.

Another approach to detecting fake information was on [28], where they detected fake LinkedIn profiles. On this occasion, the dataset used for training included real and AI-generated profiles. They tested multiple LLMs like BERT and RoBERTa, but BERT resulted in the highest accuracy of 95.67%. These investigations prove the efficiency of Large Language Models for Natural Language Processing in the misinformation field. Regardless, none of these studies addresses health-related misinformation on social media.

When combatting misinformation, the difficulty arises when determining what is spreading and how experts can correct it. To determine if a text is spreading lies, one must understand or find credible sources, such as peer-reviewed studies or expert opinions, to verify the truth. Some disinformation can be easier to identify like hoaxes, but other things, such as conspiracy theories, require more resources to debunk. When the topic of the text becomes complex or not identifiable, some credible sources help with the rebuttal. These tasks are time-consuming and require an expert in the field for accuracy. In addition, the explanation must be expressed so that any audience can

understand it. These are a few reasons that health-related misinformation is hard to combat. It requires professionals in the field to be fast at identifying misinformation and concise when correcting them. With the advancement in Artificial Intelligence, LLM can combat misinformation by classifying it and rebutting it. For the classification process, it is possible to finetune an LLM that determines if a text is misinformation. Additionally, it can generate rebuttals using RAG. With a vector database that contains peer-reviewed research, it can ensure that the information is factual. This AI approach can reduce the dependency on experts and have a system that can act in real-time to prevent a significant spread of misinformation.

2.4 Twitter Health Surveillance (THS)

The THS system classified tweets related to health issues [2]. The experiment utilized LSTM and GRU to classify tweets as being medical related, medical unrelated, or ambiguous. THS data extraction pipeline can be found on Figure 2.5. First they extracted data from the Twitter API and sent that data to an Apache Kafka queue. Then, a consumer sent it to Apache Spark to process the data and store it in a Hive warehouse. Later, a preprocessing phase for each tweet occurred, which removed hashtags, mentions, emojis, and web links. The agent trained with the resulting plain text. This version used recurrent neural network (RNN) because of its advantages with sequential data. They tested various combination architectures, but the one with the highest result was an LSTM layer, with no attention, and a GRU layer; it had an F1 score of 86%, a recall of 89%, and a precision of 83%.

Later the project was updated to find similarities between tweets [29]. This new version tested convolutional neural networks (CNN) and recurrent neural networks to classify how closely related are two different tweets. They used a ranking method that

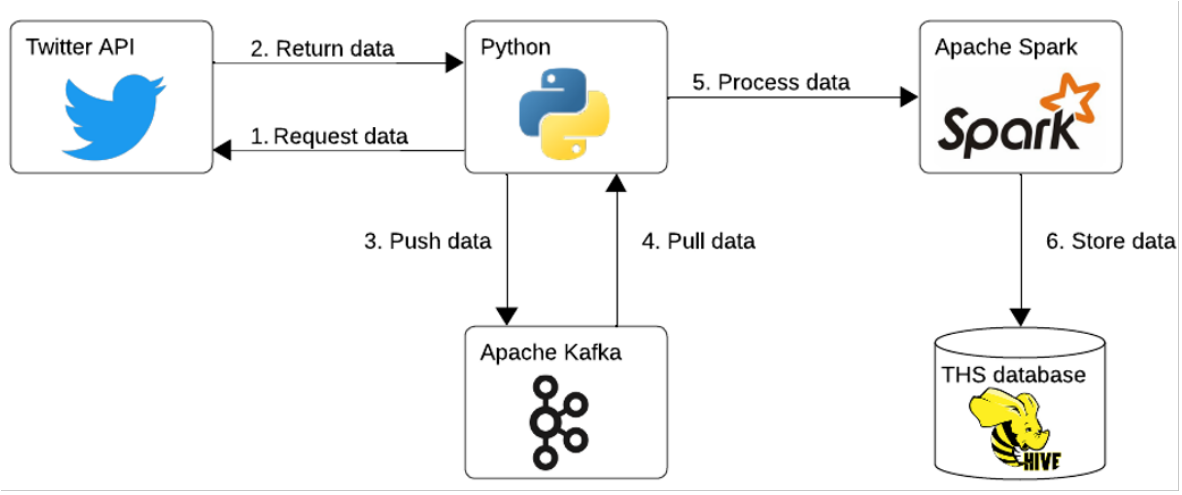

Figure 2.5: THS Architecture

gave a higher score if two randomly selected tweets were similar. The agent received an input of triplets where the first element was compared against the other two elements. The system trained on two data types: raw tweets and cleaned tweets. This latter had stop-words removed and a lemmatization process. Cleaned tweets proved a higher accuracy in the recurrent neural network for regular LSTM and bidirectional networks; in contrast, the raw tweet had a better result in CNN validation. All three models had similar results; the highest validation accuracy was the regular LSTM with 90%, followed by the other two with 87%. Nonetheless, the training time for the CNN was the fastest, with regular LSTM in second place and the slowest being the bidirectional LSTM network.

However, both of these experiments removed special characters or elements from the original text. By then, tweets had a limitation of characters, making each one of the crucial the context. Removing special characters, could remove information that the author of the text intended. We can have the example in Figure 2.6, where a based model GPT classified a text. The model uses the hashtag and mention as context to determine if the text is health related or not. This is a good presumption, but these

Id: 5
Sentence: As the use of Wolbachia infected mosquitos is approved to combat Zika in the USA how will production demands be met? https://t.co/XrbcstLMcv A nice mention for @michiganstateu in there too! #spartanswill #zika
Related: Yes
Reason: This sentence is related to a health issue. It discusses the use of Wolbachia infected mosquitos to combat Zika, which is a health concern. The mention of production demands being met suggests that this is a topic related to public health efforts and strategies. The mention of @michiganstateu and the hashtags #spartanswill and #zika further indicate the relevance to health issues.

Figure 2.6: ChatGPT Classification Example – Health Related

platforms' users could use hashtags or mentions that are not relevant to the text.

Chapter 3

System Architecture

This section presents the general architectures of the systems we used for the experiment. We illustrate the Extraction, Transformation, and Load (ETL) pipeline for the research papers we used for the misinformation rebuttal. Additionally, we have our misinformation classification process, where we classify the health misinformation and refute it. Finally, we have the UI design where users interact with the texts and the results.

3.1 Research Paper ETL Pipeline

Our model must use credible sources of information to rebut misinformation. We identified PubMed [30], an online library that contains peer-reviewed medical literature. We want to extract the papers and store them in a vector database. To extract these papers, we used the BioC API [31], which has access to the PubMed library. However, the API needs the research paper's identifier, known as PubMed Central (PMC) ID. We design a scraper to extract these identifiers from the official PubMed site. The pipeline in Figure 3.1 shows the processes of data extraction.

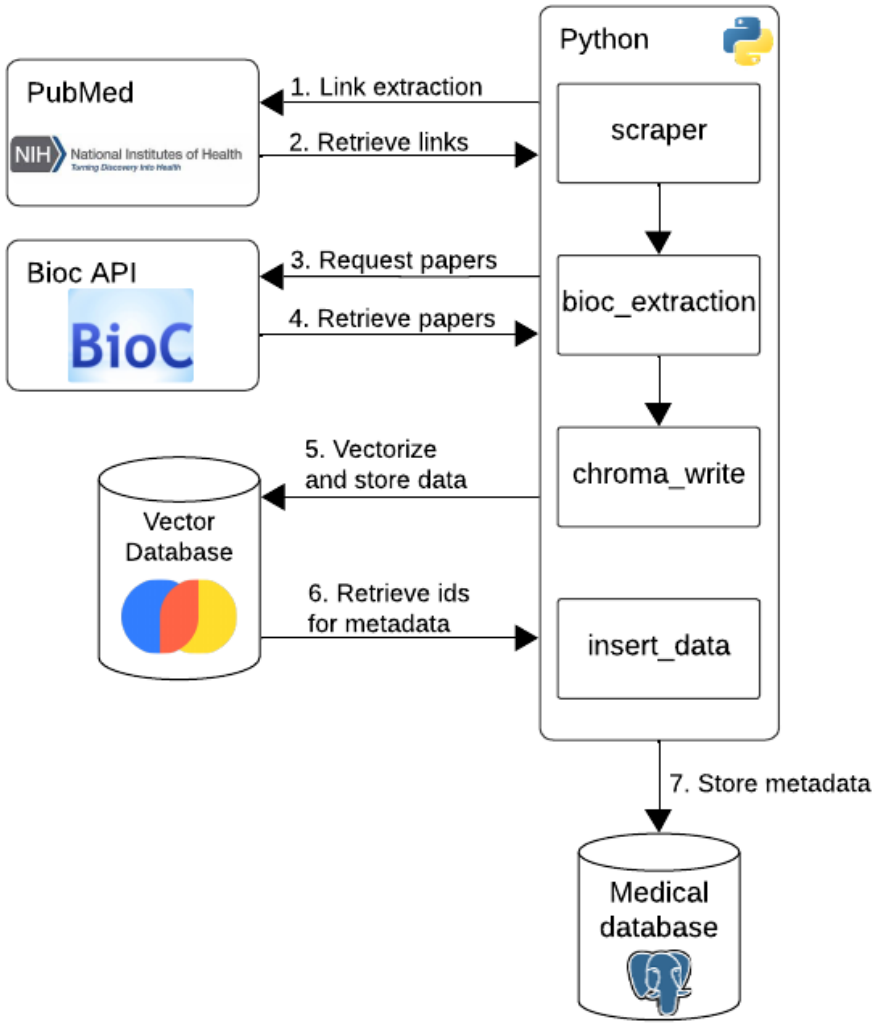

Figure 3.1: Medical Data Extraction Pipeline

3.1.1 Scraper

The first step of the pipeline was identifying what papers we needed to extract. We selected 17 topics for the data extraction process, that can be seen in Table 3.1.

To extract them, we built a scraper in Python using Selenium and BeautifulSoup libraries. We used Selenium to retrieve the web source from PubMed's website, and BeautifulSoup was used to get the links to each paper. These links contained the PMC

Table 3.1: Topics for Research Paper Extraction

Topics	
allergy	covid vaccine
bird flu	flu vaccine
cancer	headache
chickenpox	influenza
common cold	monkeypox
conjunctivitis	stomach aches
covid sickness	swine flu
covid symptoms	zika
covid treatment	

identifier. For each topic, we selected 5,000 PMC identifiers. These identifiers were grouped by topic and stored locally in Comma Separated Value (CSV) files.

3.1.2 BioC API

After retrieving those identifiers, we need to extract the research papers. Using the PubMed API, BioC, we made requests that returned the documents as JSON. These JSONs were preprocessed to contain texts, and we removed tables and figures. Later, the paper's sections -introduction, methodology, results, and others- were combined as one attribute, excluding references. We removed tables, figures, and references from the context to ensure the chunking process worked appropriately. If the data is not preprocessed, when performing RAG, we can retrieve data that is not useful. After that, we turned the result into a new JSON that contained the research metadata and its context.

3.1.3 Vectorizing data

Later, each research paper's context was split into chunks using LangChain. Then, we used an LLM, BAAI [32], to embed these chunks. A universal unique identifier (UUID) was combined with each chunk and stored in a Chroma [17] database. After uploading the data to Chroma, we added these UUIDs to their JSON.

3.1.4 Store metadata

Now, with all papers vectorized, we upload the metadata into a Postgres database. First, we validate that there are no duplicate records in the system. To prevent duplicates, we search for the paper's reference. If any is found, we delete the chunks from the vector database. Additionally, any research that did not contain at least an abstract was removed. That ensures that there is no repetition or inconsistency when doing the rebuttal. Later, we upload this data into the system following the schema found in Figure 3.2. The tables in this schema are as follow:

Research: The table contains the research paper data. Its attributes are title, which is the research paper title; context, the paper's text; paper_ref, the complete reference of the paper, used to prevent duplicates; and fullpaper, which is a boolean that is true if the paper contains an abstract, introduction, methodology, discussion, conclusion, and references.

Chunks: This table pairs the UUIDs from the paper's chunks and their respective research record.

Keyword: Some research papers contain keywords that allow the reader to know the subjects mentioned in the paper.

Author: Stores the first and last names of all authors identified in the research paper.

Reference: All references that are present in the research paper.

Topic: This contains the different topics used to search the papers.

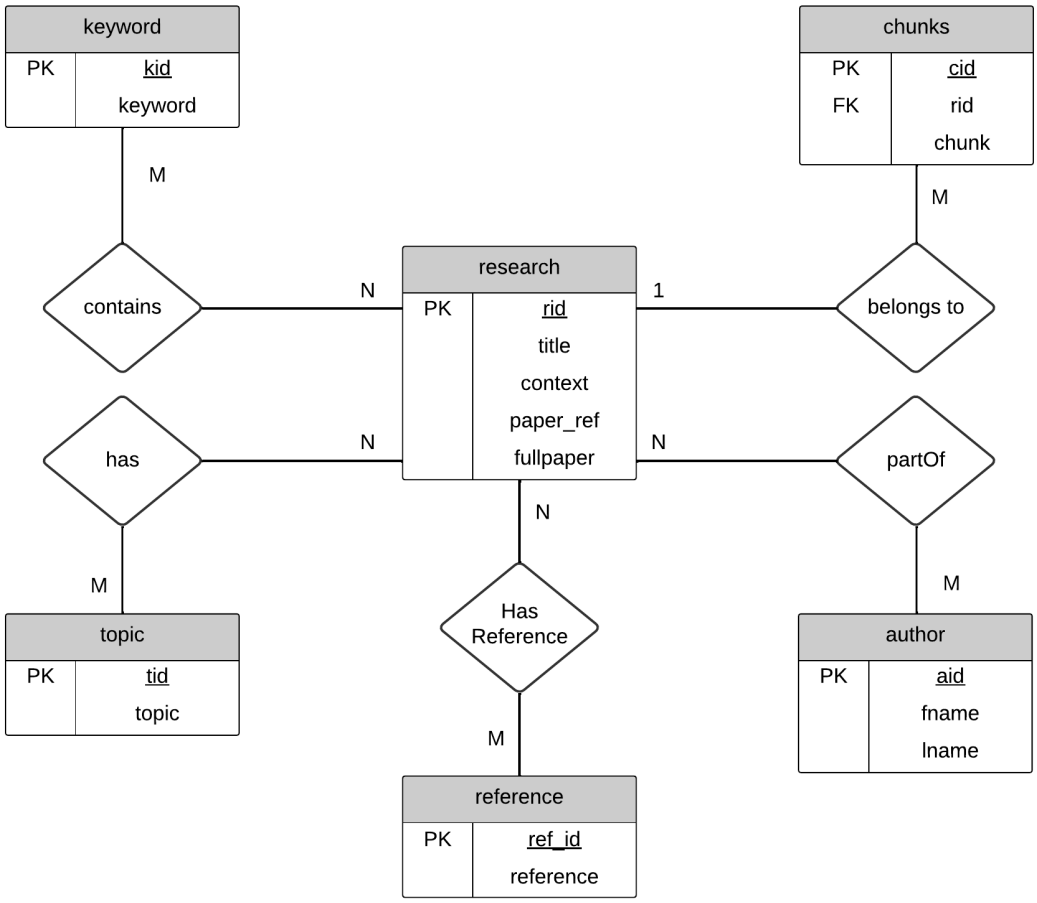

Figure 3.2: Research Papers Schema Diagram

We started the search with 85,000 peer-reviewed papers. After finishing the filtering and data cleaning, we ended with 56,365 different research papers.

3.2 Misinformation Rebuttal Pipeline

After training the models and storing the context for the rebuttal, we create the model pipeline. The pipeline shown in Figure 3.3 shows the process of receiving a text,

making the classifications, and returning an explanation of why it is misinformation.

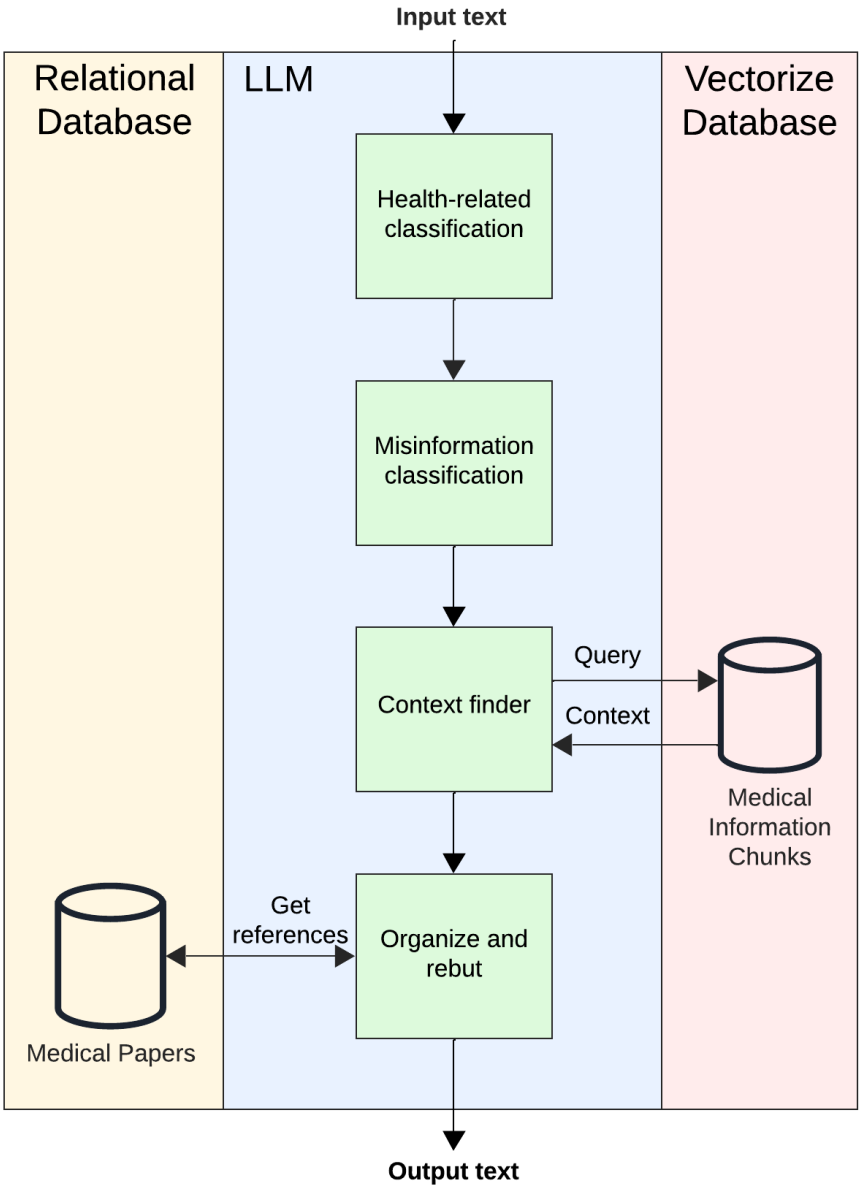

Figure 3.3: Misinformation Rebuttal LLM System Architecture

3.2.1 Health-related Classification

The first part of the pipeline is determining if the text is related to health. When the system receives the input text, it sends it to a model finetuned to classify health-related

texts. This model determines whether the input is related, unrelated, or ambiguous. If the text is health-related, we continue to the next part of the pipeline. When the text is non-related or ambiguous, we end the process. We end this because other topics are out of the model's scope.

3.2.2 Misinformation Classification

After confirming that the text is health-related, we want to validate that it contains misinformation. The finetuned model can only return two possible results: misinformation or non-misinformation. If the model finds no misinformation after evaluating the text, the final output will return both classifications. When the result contains misinformation, we start the search for research papers.

3.2.3 Context Finder

Before starting the search on the vector database, we must understand the topic of the text. To automate this process and generate a query as precise as possible, we use Ollama [33]. That tool allows us to execute locally a pre-trained LLM that generates text. Thus, we send a query to Ollama asking it to make a one-sentence query for the vector database related to the input.

Input

```
#nih fauci, @cdcdirector @sgottliebfda &amp;amp; @barda bright
expected to be grilled tomorrow over ineffective #flu vaccine
at @housecommerce #suboversight bets on fauci saying universal
#influenza vax in 5, maybe 10 years? my preview here: https:
//t.co/fsqefwhik7 #cdc #fda #vaccines
```

Output

Flu vaccine effectiveness and future universal influenza vaccination strategies.

The above example shows that Ollama can identify the topics of the original text. That tweet mentions that the flu vaccine is ineffective and asks how long it will take for a universal influenza vaccine. The LLM identified the topic and made a query that can help find research papers related to it. However, it is essential to mention that the LLM output can vary. As mentioned previously, this is a statistical model and can have slight variations in its result. Now, this output is sent to the Chroma database to retrieve chunks of research papers. For this experiment, our model returns eight chunks that are closest in context to the query. We selected this number because it returns an appropriate amount of context without Ollama truncating the text. These chunks are then sent to another model to be analyzed and organized.

3.2.4 Organize and Rebut

The final part of our pipeline is using RAG to provide an answer that explains why the original text is misinformation. First, we retrieve the references of the chunks we use for the context. Then, we send the original text with the chunks, as context, to Ollama for the model to evaluate. Ollama returns a 2-3 sentence result that explains the text's misinformation. The final output is a JSON with the following attributes:

chroma_value: If the text is health-related and is misinformation, this will have the rebuttal output from Ollama, consisting of 2-3 sentences. If the text is non-related or non-misinformation, it will give a default value saying so.

health: This is the input text health classification.

misinformation: This is the input text misinformation classification.

references: This attribute stores the list of references used for the rebuttal.

t_context: The original text.

The pipeline enables us to automate the classification process and rebut misinformation using peer-reviewed research. By leveraging fine-tuned models, vector search, and RAG, the architecture provides concise, fact-based responses. A key advantage of this approach is its ability to explain complex content accessible to non-technical readers, giving them a clearer understanding of false information. This can assist professionals in the field to mitigate the spread of lies that can negatively impact public health.

3.3 User Interface (UI)

With the pipeline correctly working, users need to interact and view the results. We created an user interface were they can interact with texts and see the classification process. Figure 3.4 shows the main menu of the interface. When we select a text a side screen appears that shows the results from the misinformation classification pipeline.

Figure 3.5 presents a case where the models classified the text as health-related and misinformation. The top of the side screen shows the text that was classified. Number 1 shows the health classification, and number 2 shows the misinformation classification. Below that is the rebuttal generated by the models. Finally, we include the references of the research used for the rebuttal. An alternative view of the side screen is Figure 3.6; when it does not find misinformation, the model does not generate an answer nor cite any research.

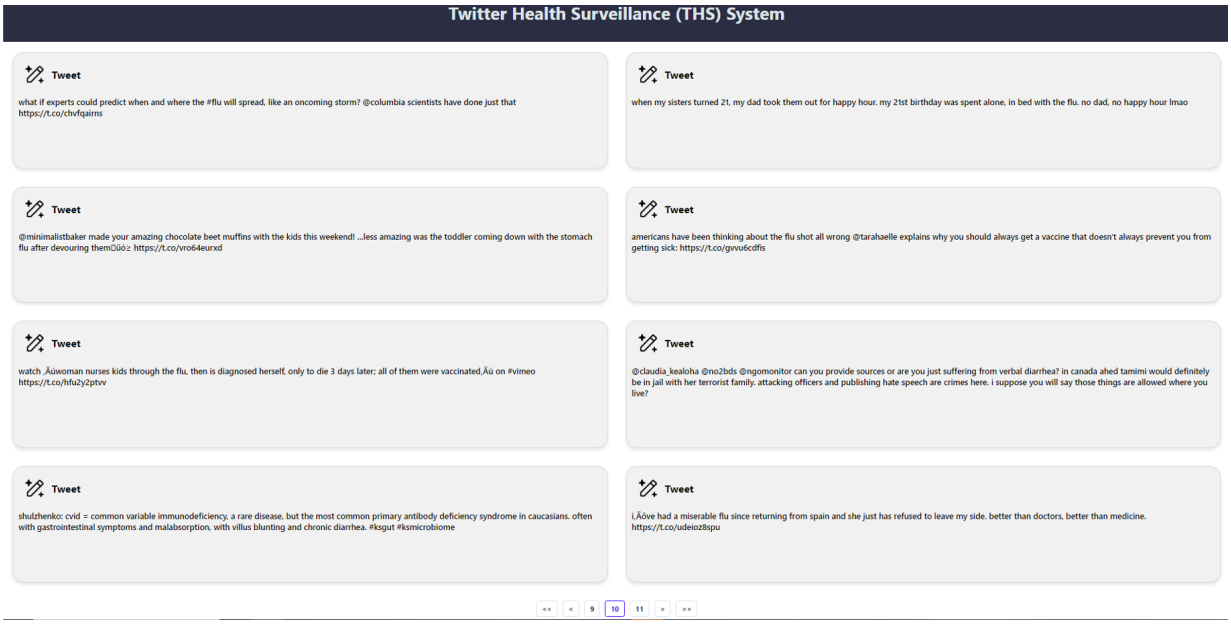

Figure 3.4: THS Frontend - Menu

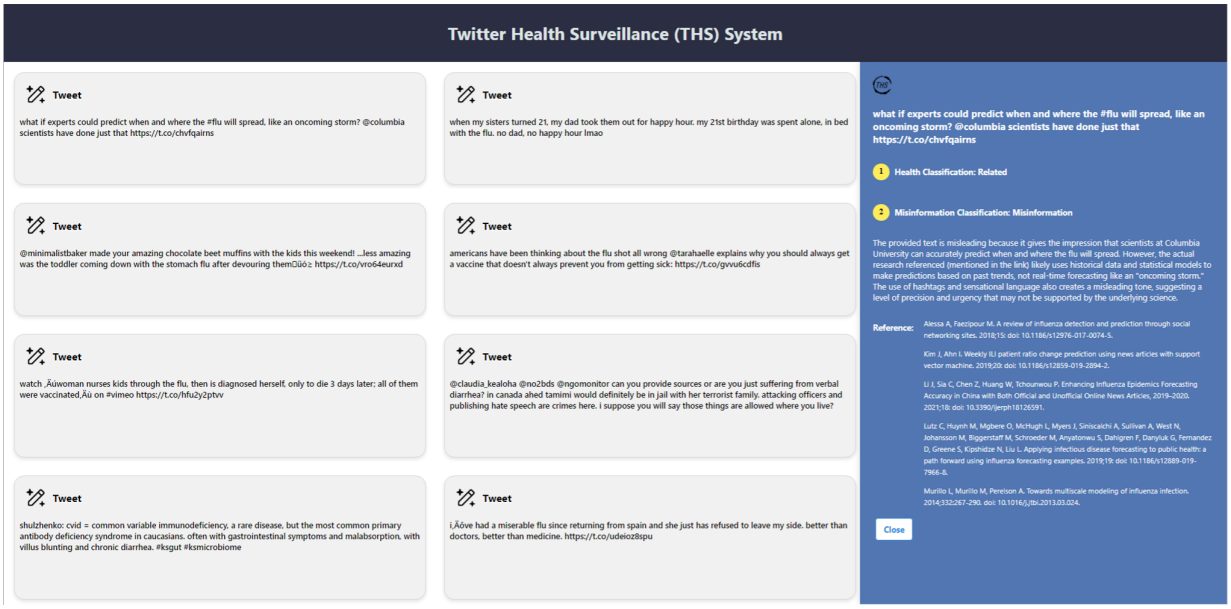

Figure 3.5: THS Frontend - Misinformation Classification

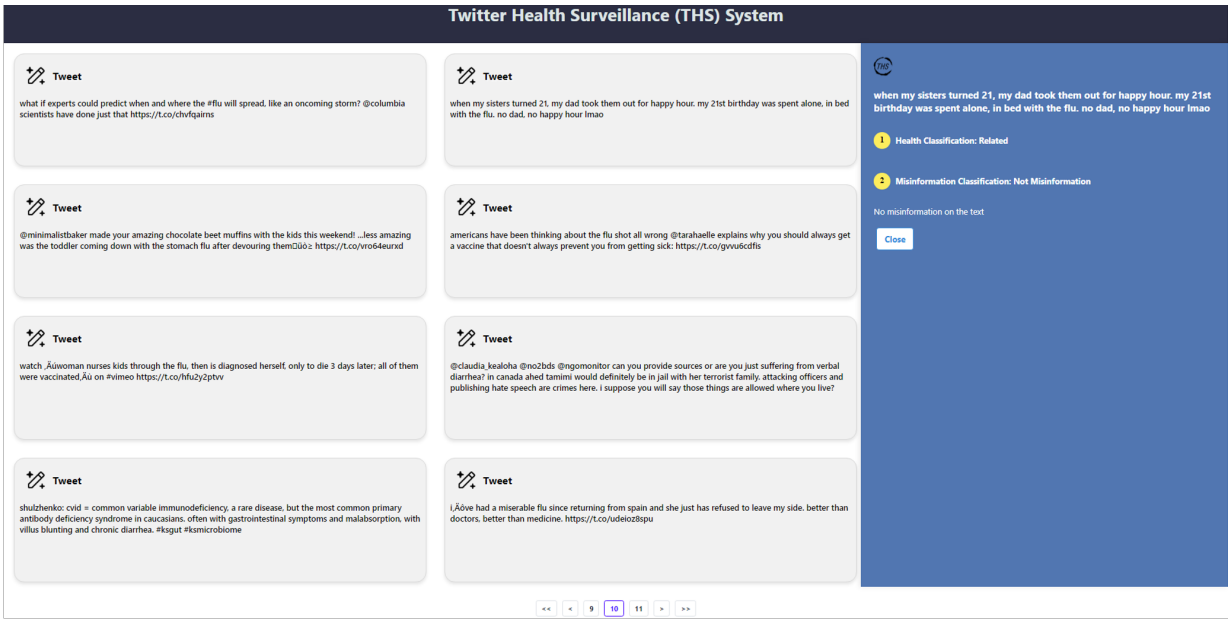

Figure 3.6: THS Frontend - No Misinformation Classification

Chapter 4

Experimental Methodology

4.1 Performance Metrics

To evaluate the models effectiveness we used the following metrics:

- **Precision**: This metric evaluates the proportion of positive that the model classified as positive that are actually positive.
- **Recall**: This metric evaluates the proportion of the positives that the model classified correctly again all positives.
- **F1**: This metric balances the precision and recall scores.
- **Elapsed Time**: To evaluate the models training time.
- **BERTScore**: Compute a similarity score between two different sentences, to verify that the rebuttal generated by the LLM relates to the classified text. [34].

4.2 Hardware

This experiment was implemented in a cluster. Only one node was used for the finetuning and to host the application. The hardware specification for the node is found on Table 4.1.

Table 4.1: Cluster's Node Specifications

Hardware	**Description**
Operating System	Ubuntu 20.04.2 LTS
Hard Disk	250GB
RAM	87.9GB
Processor	Intel(R) Xeon(R) Silver 4214 @ 2.20GHz
GPU	NVIDIA TESLA V100s 32GB

4.3 Software

There were various software tools used for the project. The training, ETL pipeline, and REST API were implemented with Python 3.9.19. To finetune the models we used PyTorch 2.0.1 with GPU support enabled for NVIDIA CUDA 11.7. To initialize and use the models we implemented the Transformers 4.34.0 library. The models were imported from Hugging Face (HF) [35]. We used BERT, T5, and LLaMa-2 for our research. To reduce the model memory usage, we use the PEFT 0.12.0 and bitsandbytes library for LoRA. In Table 4.2 we see the specifications for each model. All these models are base model, which means they are not train to accomplish a specific task. Also, we use Postgres 14 to store our extracted research papers. In addition, we implemented Chroma 0.4.24 as our vector database. Next, for the RAG process, we installed Ollama [33] tu run LlaMa3.1 8B [9] parameter model. Finally, to create the UI to show the results we used React.

Table 4.2: LLM Specifications

Model	HF name	Architecture	Parameters
BERT	bert-base-uncased	Encoder Only	110M
T5	t5-base	Encoder-Decoder	220M
LlaMa-2	Llama-2-7b-hf	Decoder-only	7B

4.4 Datasets

This section describes the health-related and misinformation datasets used for training, their structure, and the preprocessing techniques applied.

Table 4.3: Training Datasets

Dataset	Category	Label	# of records
Health Related Dataset	Unrelated	0	3828
	Related	1	7848
	Ambiguous	2	765
Misinformation Dataset	Misinformation	0	3638
	Not Misinformation	1	5111

4.4.1 Health-related Dataset

The health-related dataset comprises over 12,441 tweets extracted from the previous THS project. Health professionals labeled the tweets in this dataset as related, unrelated, or ambiguous. As shown in Table 4.3 this dataset is imbalanced; thus, we applied class weights to the loss function. The imbalance can cause the model to predict the most frequent class disproportionately, reducing its ability to generalize.These weights assign higher penalties to errors in underrepresented classes. The weights prevent the models from overfitting, mitigating the effects of class imbalance during training. The weight penalty for each class is shown in Table 4.4.

Table 4.4: Health-Related Dataset Weight Penalty

Category	**Weight Penalty**
Unrelated	3.25
Related	1.58
Ambiguous	16.26

4.4.2 Misinformation Dataset

The misinformation dataset, with 8,749 texts, combines data from different sources such as news, social media, and blogs. These records have two labels: misinformation and non-misinformation. As in the health-related dataset, we also applied class weights because of the imbalance, as Table 4.3 shows. The weight penalties used for the imbalance is shown in Table 4.5. Multiple datasets from different sources were combined for this experiment. The dataset in [36] consists of social media texts that mention the monkeypox. We only used "text" and "binary_class" columns from that dataset. Another source was [37], which classified Covid-19 texts. That dataset contains articles and posts from different online sources. They labeled the data based on official sources that validated the text claims. In this dataset, we only used the "title" column and the label assigned to the file. We used that dataset because, occasionally, online users share news reports or articles based on the headline alone. Next, the last dataset [38] data, comes from a variety of sources. The dataset only contains misinformation records such as posts, news, videos, emails, messages, and others. We selected English-only records from that dataset because BERT is limited to this language. Now, we used only the "Title", "Narrative_Description" columns. These two columns were appended as one string for the model to train.

Table 4.5: Misinformation Dataset Weight Penalty

Category	**Weight Penalty**
Misinformation	2.41
Not Misinfomation	1.71

4.4.3 Data Preprocessing

Before we start fine-tuning the models, we preprocessed both datasets. As initially mentioned, we want as much context as possible for our model to train. That meant the classification process would include links, mentions, and hashtags. These elements are important for context because users can convey sentiments that could be relevant to text. However, these elements contain various characters, and the embedding might struggle with out-of-vocabulary or irregular patterns. Thus, we opted to use special tokens to replace them. We use special tokens to replace elements with patterns that do not contribute to semantic meaning, such as random strings in URLs. For example, some URLs are strings with random values, which can result in the embedding assigning the wrong values. Because of this, we replace these items with their respective tokens. Any URL was replaced by *[LINK]*, mentions by *[MENTION]*, and hashtags by *[HASHTAG]*. That allows us to keep the elements and give them some importance in the text. An example of this is shown below.

Example 1:

Original Text

```
listening to these amazing infectious disease experts talk
about #influenza at the @nmnh/@asmicrobiology #flu program like
https://t.co/ehnpz6nsyg
```

Preprocessed Result

listening to these amazing infectious disease experts talk about [HASHTAG] at the [MENTION]/[MENTION] [HASTAG] program like [LINK]

4.5 Fine-tuning

The LLMs used for this paper are pre-trained, meaning that the model learned how words relate to each other. In other words, the model is trained to understand a language. This process is computationally expensive and requires a large amount of data. Instead of creating a model from scratch, we can teach an existing model to learn a new task such as text classification, translation, generation, or other. That process is called fine-tuning. Fine-tuning is when an existing model is trained to accomplish a task. In this case, we taught the models to classify two types of texts: health-related and misinformation-related texts.

We identified three models to perform these classification tasks. The models used for this experiment are Bert, T5, and LLaMa-2 in their base form. Each one has a different architecture, as shown in Table 4.2. The architectures have their specialty; thus, we performed two processes of fine-tuning: sequence classification and CLM. BERT and LLaMa-2 were only trained on sequence classification, while T5 was trained on sequence and CLM. T5 was the only model capable of that because BERT encoder-only architecture does not allow text generation, and LLaMa-2 required more resources than the ones available. Additionally, these models have large numbers of parameters. That can be a drawback when fine-tuning because the process requires excessive resources.

4.5.1 Memory Usage Estimation

Our initial approach for the experiment was to use the model with default parameters. That meant using the base model with Adam optimizer and with a floating point of 32 (fp32). However, that was not possible because the model requires an excessive amount of memory. To estimate the resources that each model requires, we use the formula in [39]. Their formula provides a result with an overhead of around 20%.

$$Memory_{model}, fp32 = (4bytes/param)(No.param) \tag{1}$$

For the example, we use T5, which has 220M parameters. If we use Equation 1, T5 requires 839MB to initialize.

$$Memory_{optimizer} = (12bytes/param)(No.param) \tag{2}$$

Then, we calculate the optimizer state. We consider AdamW because it is the most common optimizer. When using Equation 2 we get that the optimizer for T5 requires 2.46GB.

$$Memory_{gradients} = (4bytes/param)(No.param) \tag{3}$$

Next, we have the gradient. The gradient memory usage is similar to the model. Equation 3 shows that it requires 839MB for T5.

$$Memory_{activation} = sbhL(10 + \frac{24}{t} + 5\frac{a \cdot s}{h \cdot t}) \tag{4}$$

The final attributes that we need to consider are activation and batch size. We estimate the variables for Equation 4 based on the experiments conducted. The first element is **s**, which refers to the token length sequence. Then, the **b** is for the batch size.

The **h** stands for the model's hidden size and dimensionality. Now, **L** is the number of layers in the model. Next, we have **a** for the attention head of the model. Lastly, we have **t** for parallelism. The variables for each model can be found in Table 4.6. Finally, if we calculate T5 activation memory usage, it results in 3.80GB.

Table 4.6: Models Activation Parameters

Model	**Sequence (s)**	**Batch Size (b)**	**Hidden Size (h)**	**Layers (L)**	**Attention Head (a)**	**Parallelism (t)**
BERT	256	16	768	12	12	1
T5	256	16	768	12	12	1
LLaMa-2	256	16	4096	2	32	1

With all these values calculated, we can estimate the memory usage per model using Equation 5. The results for each model are presented in Table 4.7. These estimates show the minimum memory required for each model. For BERT we need 3.95GB, T5 is 6.00GB, and LLaMa-2 is 152.39GB. Knowing that our cluster only has a GPU with 32GB, it would not be possible to fine-tune LLaMa-2. However, there is an alternative that reduces the model size.

$$TotalMemory = Model + Optimizer + Gradients + Activation \tag{5}$$

Table 4.7: Models Memory Usage Estimate (MB)

Name	**Parameters**	**Model**	**Optimizer**	**Gradient**	**Activation**	**Total**
BERT	110M	419.62	1258.85	419.62	1944.00	**4042.09**
T5	220M	839.23	2517.70	839.23	1944.00	**6140.16**
LLaMa-2	7B	26702.88	80108.64	26702.88	11264.00	**156042.40**

4.5.2 Parameter-Efficient Fine-Tuning (PEFT)

PEFT is a library used to reduce the parameters of a model that will be fine-tuned. The library allows us to use the Low-Rank Adapter (LoRA). That adapter helps us reduce the model's memory usage. The main reason is that we can initialize the model in 8 bits instead of 32 bits. Also, because we have less precision, we can change the optimizer to adamw_8bits. Additionally, we do not fine-tune the entire model; we only modify a small percentage.

To determine how many parameters will be modified, we need to understand the hyperparameters of the LoRA configurations. Based on the research [14], we can estimate the number of trainable parameters. Equation 6 can give the approximate trainable parameters. The variable $\mathbf{d_{model}}$ is the hidden layer of the model. Then, $\mathbf{r}$ is the low-rank factor, which is how many parameters we train. The last attribute is the $\mathbf{L_{weight}}$, the number of weight matrices. That value is the product of the number of layers by the projection matrices. In our experiment, we use two model matrixes, the query matrix (Q) and the value matrix (V). The first represents what the query token is looking for in other tokens. The value matrix contains the information associated with each token; for more details about these matrixes, refer to [5].

$$Trainable\ Parameter = 2 \cdot d_{model} \cdot r \cdot L_{weight} \tag{6}$$

Currently, we can estimate the LoRA layer for each model. The hyperparameters and total trainable parameters of our experiment can be found in Table 4.8.

Table 4.8: LoRA Hyperparameters & Total Trainable Parameters

Name	$\mathbf{d_{model}}$	**r**	$\mathbf{L_{weight}}$	**Total Parameters**
BERT	768	16	24	**589,824**
T5	768	16	24	**589,824**
LLaMa-2	4096	16	64	**8,388,608**

Now, we can estimate the new memory usage that each model will require. To ensure that we can use all models, we made some changes to the parameters. First, we reuse the batch size and token sequence from 4.6. Second, we have adamw_8bits as our optimizer. Finally, we will use the alternative equations from [39] for the estimate. This time, LLaMa-2 will be the model for our calculation.

$$Memory_{model}, int8 = (1bytes/param)(No.param) \tag{7}$$

For Equation 7, we use the total number of parameters. We do this because we need to initialize the entire model even if we train just a few parameters.

$$Memory_{optimizer} = (6bytes/param)(No.param) \tag{8}$$

For the optimizer, we use 8bit_AdamW because it retains AdamW's advantages but leaves a smaller memory footprint. In this case, we only consider the number of parameters to be trained. With Equation 8 and Table 4.8, we calculate that the optimizer for LLaMa-2 requires 48MB.

$$Memory_{gradients} = (1bytes/param)(No.param) \tag{9}$$

Next, we have the gradient. The gradient consumes a similar amount of memory as the model. Equation 9 shows that LlaMa-2 requires 6675MB.

The last element is the activation, and we reuse Equation 4. The variables that the equation requires are in Table 4.6. Moreover, if we calculate LLaMa-2 activation memory usage, it results in 11GB. The new memory usage for each model is in Table 4.9. That table shows that BERT requires 1.06GB, T5 is 1.16GB, and LLaMa-2 is 17.58 GB. Those results show that it is possible to train the models on our hardware without limitations. However, these estimates can vary depending on the tasks or the model's initialization method. Causal Language Modeling can require more resources than the estimate because we turn the labels' data into a token sequence. Other factors, such as dropout, the weight we give to each LoRA parameter, and the weight decay, also affect this result. Nonetheless, this approximation shows that it is possible to fine-tune the models.

Table 4.9: PEFT Model Memory Usage Estimate (MB)

Name	**PEFT Param.**	**Model**	**Optimizer**	**Gradient**	**Activation**	**Total**
BERT	589,824	104.90	3.38	1.13	972.00	**1081.40**
T5	589,824	209.81	3.38	1.13	972.00	**1186.31**
LLaMa-2	8,388,608	6675.72	48.00	16.00	11264.00	**18003.72**

4.5.3 Training Parameters

After validating that we can fine-tune the models with our hardware, we must select the hyperparameters for our training. We present the parameters used in Table 4.10, and the description for each are as follows::

- **Learning Rate**: Step size that the model takes to adjust its parameters.
- **Batch size**: Number of training samples used in one iteration or step.
- **Epochs**: Number of times the model iterates the entire training data.

Table 4.10: Fine-tuning Hyperparameters

Parameter	**value**
Learning Rate	5E-6
Batch size	16
Epochs	20
Gradient Accumulation	8
Weight_decay	0.1
Evaluation Step	50
Evaluation Batch	2
Evaluation Accumulation	16
Warm-Up	450
Metric	f1

- **Gradient Accumulation**: Accumulation of multiple batches results during training to perform a weight update; this reduces memory usage.
- **Weight_decay**: A regularization method used to help the model generalize.
- **Evaluation Step**: Number of update steps between two evaluations.
- **Evaluation Batch**: Number of samples used in one iteration or step during evaluation.
- **Evaluation Accumulation**: The accumulation of multiple batches results during evaluation to perform a weight update; this reduces memory usage.
- **Warm-Up**: Number of warm-up steps for the learning rate.
- **Metric**: The metric used to select the best model during the evaluation process.

After setting each parameter for the training process, we must validate that the model training was correct. To achieve that, we need to split the dataset. We split the dataset for each classification task as follows:

Training dataset (70%): the data used to adjust the models' parameters during the fine-tuning training process.

Evaluation dataset (15%): the data used to validate the models' performance during the fine-tuning evaluation process. When each evaluation step finishes, we save the configuration for the model with the highest metric result.

Test dataset (15%): we use this data to select the best model between a collection of trained and validated models.

After finishing the data preprocessing and configuring the model we ensure that the rebuttal aligns with the original question.

4.6 Rebuttal Evaluation

The last part of the pipeline is to explain why a text contains misinformation. The text generated by the LLM must be on par with the question. In other words, the answer must related to the input. To validate that the input and output relate to each other, we use the BERTScore [34]. That metric uses a BERT or RoBERTa model to evaluate how semantically related two texts are. The BERTScore returns three different values: precision, recall, and F1. The precision refers to how much of the generated response is relevant to the input. The recall is how well the generated response captures the input's meaning. Finally, F1 balances both scores. In our experiment, we will focus on F1 in the BERTScore because we want our model output to capture the relevance and meaning of the input.

We tested two models for the BERTScore GPT-3.5 and LLaMa-3.1. We use the first model through the OpenAI [40] API. The second model is used locally with Ollama [33]. We tested these models because the first is a cloud model frequently used for text

generation like ChatGPT, while the second is cost-effective. Both use RAG to generate a response based on the provided research papers.

For the evaluation process, we selected 200 random texts from health-related and misinformation-related datasets. These texts were passed through the misinformation pipeline to be classified by our fine-tuned models. The pipeline identified 71 texts out of the 200 that match both classification types. Then, we generated a rebuttal and compared it with the original text using BERTScore. Finally, with all the setup done, we started gathering metrics.

Chapter 5

Performance Evaluation

This section presents the results of our models on the classification tasks. The results show each model's precision, recall, f1 score, training time, and BERTScore.

5.1 Health-Related Classification

We present the results of the health classification process and compare them with the best overall model of the previous THS project. They concluded that their best model is an LSTM, with no attention, and a GRU layer.

5.1.1 Precision

Table 5.1: Health Related Precision Result

Model	**Result**
BERT	0.85
LLaMa-2	0.94
LSTM GRU NO ATTENTION	0.83
T5 (Causal)	0.85
T5 (Sequence)	0.48

Table 5.1 shows the result for the precision metric for the related classification. For clarity, we focus on this class because our project goal is to detect health-related misinformation. The best-performing model here was the LLaMa-2 model, which has a precision of 94%. Tied for second place are BERT and T5 (Causal), with 85% precision. Next is the THS model, with 83% precision. Lastly, we have a T5 (Sequence) model with a relatively low precision of 48%.

5.1.2 Recall

Table 5.2: Health Related Recall Result

Model	**Result**
BERT	0.91
LLaMa-2	0.84
LSTM GRU NO ATTENTION	0.89
T5 (Causal)	0.95
T5 (Sequence)	0.44

Table 5.2 shows the result for the recall metric for the related classification. When comparing this metric with the THS investigation, there is a noticeable difference. Their results show that only the LSTM layer, no attention, and a GRU layer model was the only one with a result over 80%. In contrast, most of our models had a score of at least 80%. Comparing this result with the precision table (Table 5.1, our model did not score drastically lower. Here, our best model was T5 (Causal), with a performance of 95% in recall. Next, we have BERT with 91%, followed by the THS model with 89%. Then, the model with the highest precision, LLaMa-2, ended with 84%. Lastly, we have again the T5 (Sequence) model with a 44%, lower than its precision.

5.1.3 F1

Table 5.3: Health Related F1 Result

Model	**Result**
BERT	0.88
LLaMa-2	0.89
LSTM GRU NO ATTENTION	0.86
T5 (Causal)	0.90
T5 (Sequence)	0.46

Table 5.3 shows the result for the F1 metric for the related classification. The F1 score is the balance between precision and recall. When we compare the results with the THS model, most models have a higher F1 score. The results show that T5 (Causal) had 90%, LLaMa-2 with 89%, BERT ended with 88%, and T5 (Sequence) scored a 46%. In contrast, the THS model had an 86%, ending in second to last place.

5.1.4 Training Time

In Figure 5.1, we present our model's training time. As mentioned earlier, we trained each model for 20 epochs with a batch size of 16. BERT trained faster than any other model, which took 187 minutes. The T5 (Sequence) model took 1442 minutes, while the T5 (Causal) model finished in 1569 minutes. Lastly, LLaMa-2 took 5126 minutes to train. Based on these results, we can infer that models with fewer parameters train faster. BERT trained 27.4x faster than LLaMa-2.

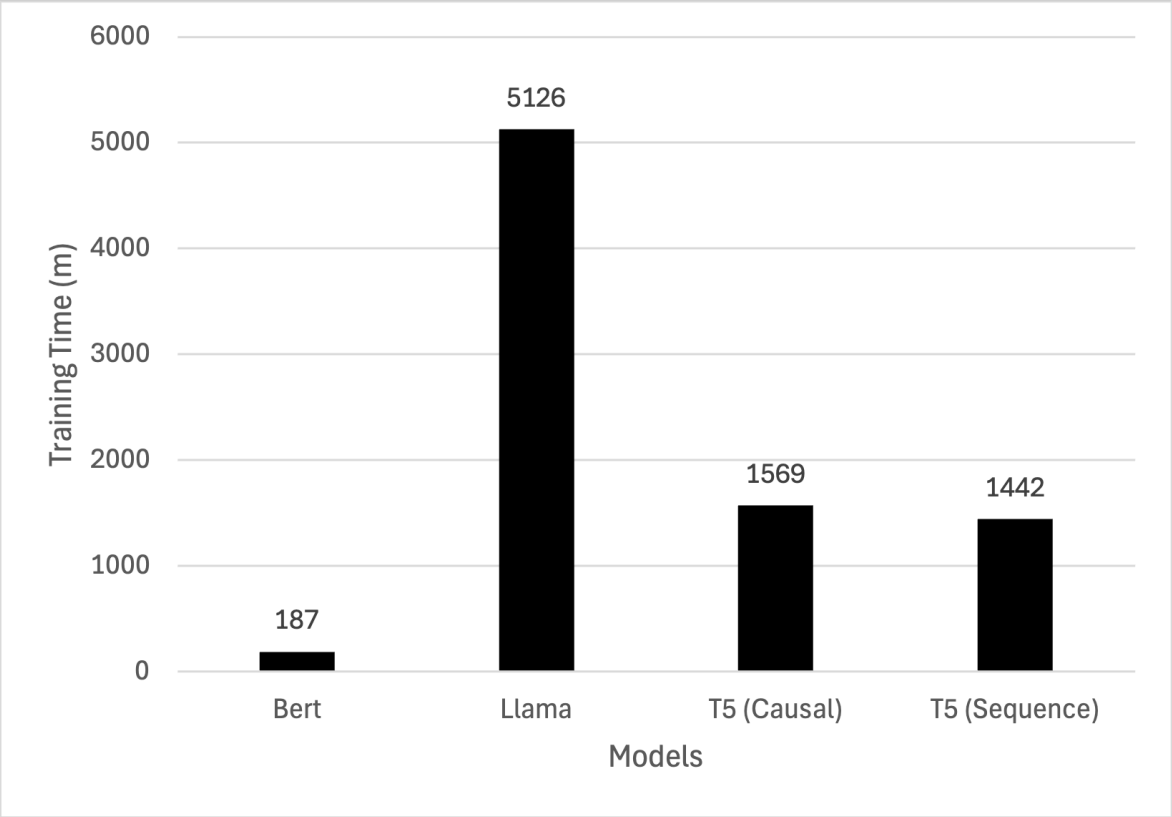

Figure 5.1: Health-Related Models Training Time

5.2 Misinformation Classification

In this section, we present the misinformation classification results. However, in this section, we did not compare with the THS model because they did not train a model to classify misinformation.

5.2.1 Precision

Table 5.4: Misinformation Precision Result

Model	Result
BERT	0.90
LLaMa-2	0.98
T5 (Causal)	0.99
T5 (Sequence)	0.99

Table 5.4 shows the result for the precision metric for the misinformation classification. In this case, we focus on the misinformation class because it is our project goal. Our best-performing models here were both T5 models, which have a precision of 99%. The next model with the highest precision was LLaMa-2, with a score of 98% precision. The lowest-performing model was BERT, which resulted in 90% precision. Nonetheless, all models had a score of at least 90%.

5.2.2 Recall

Table 5.5: Misinformation Recall Result

Model	Result
BERT	0.94
LLaMa-2	0.95
T5 (Causal)	0.92
T5 (Sequence)	0.85

Table 5.5 shows the recall metric's results for the misinformation classification. When we compare these results with the precision table (Table 5.4), our model maintains a similar score. Here, the model with the best results was LLaMa-2, with a performance of 95%. Next, we have BERT with 94% and T5 (Causal) with a 92% recall. Finally, T5 (Sequence) ended with 85% in the recall.

5.2.3 F1

Table 5.6: Misinformation F1 Result

Model	Result
BERT	0.92
LLaMa-2	0.97
T5 (Causal)	0.96
T5 (Sequence)	0.92

Table 5.6 shows the result for the F1 metric for the misinformation classification. This result balances precision and recall. The results show that the model with the best F1 was LLaMa-2 with a 97%. Next is T5 (Causal) with a 96% F1, and tied to last, we have BERT and T5 (Sequence) with 92%. All models had an optimal F1 score over 90%.

5.2.4 Training Time

In Figure 5.2, we present the training time for the misinformation models. We trained the models using the same parameters as in the Health-Related classification. When we compare these results with the Health-Related (Figure 5.1), the average training time was less. A possible reason is that we used less data for the training. The only exception was BERT, which took 351 minutes to train. Next is T5 (Causal) with 371 minutes and T5 (Sequence) with 1274 minutes. Finally, LLaMa-2 took 2840 minutes to train. A noticeable difference was T5 (Causal), which trained 4.23x faster for this dataset compared to the health-related dataset. Nonetheless, BERT finished the fastest, being 8.10x faster than LLaMa-2.

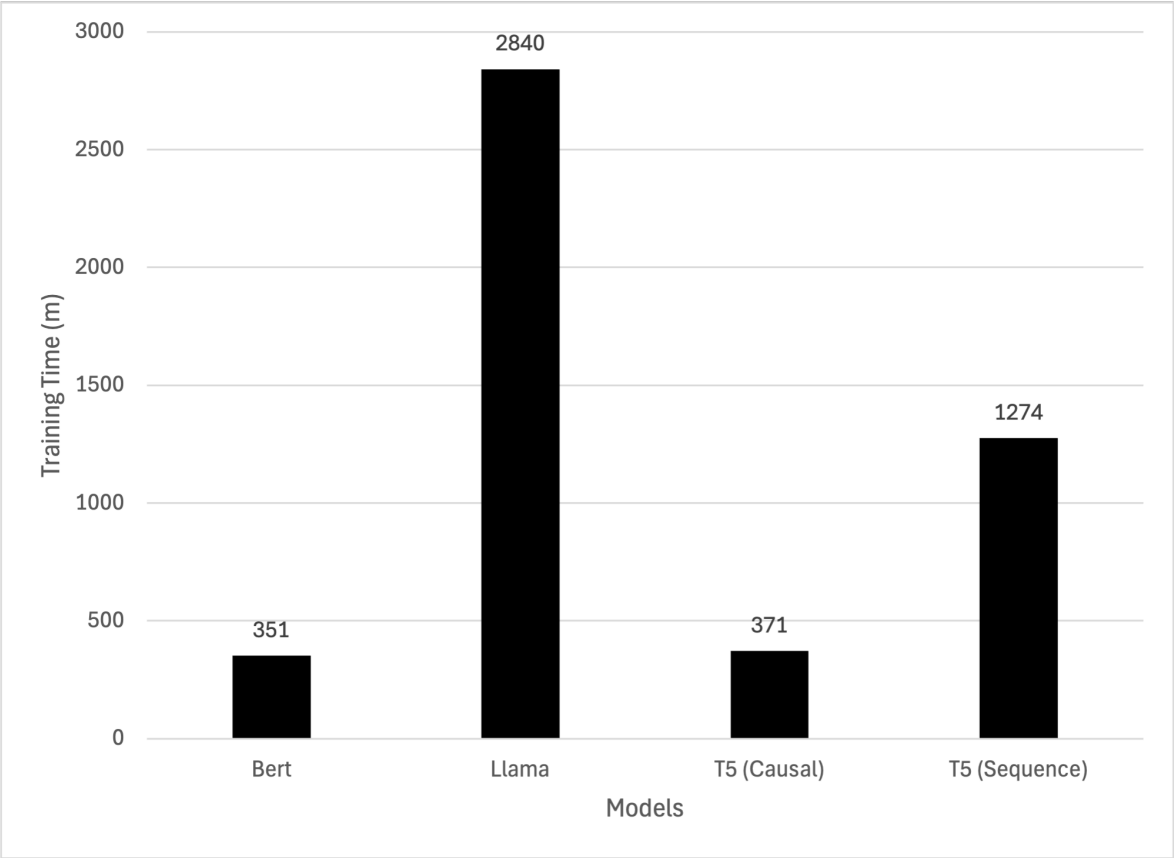

Figure 5.2: Misinformation Models Training Time

5.3 BERTScore

Our model BERTScores' F1 is shown in Table 5.7. We calculate these values by taking the average score of the 71 texts classified as health-related and misinformation. The table shows that both models, had a 82% F1 score. That means that the generated output are closely related but not exactly the same.

Table 5.7: BERTScore F1 Results

Model	**Result**
LLaMa-3.1	82%
GPT-3.5-turbo	82%

5.4 Discussion

The results present that most LLMs outperformed the THS model. We applied preprocessing techniques such as replacing links, mentions, and hashtags with special tokens. The training parameters for all models were 20 epochs, batch size of 16, 8bit initialization, and adamw_8bit optimizer. Additionally, the LoRA hyperparameters were 16 for the rank, an alpha of 32, a dropout of 0.05, and a bias for all parameters. The score metrics we used for the model evaluation were precision, recall, and F1. In our case, we want to focus on F1. That metric helps us reduce false negative results but not overclassify false positives.

Our model with the best result for the health-related classification (Table 5.3) was T5 (Causal), with a 90% F1 score, while the THS model had 86%. However, the trade-off for this model is that the training is computationally expensive. The reason is that labels in T5 (Causal) are text instead of numbers. Those texts must be embedded, which requires more processing power. Also, that model cannot have class weights because of the structure of the embedding. Additionally, the training time (Figure 5.1) for T5 is more extensive when compared to BERT, 8.4x. Now, BERT had a slightly lower result with 88%. Nonetheless, when we factor in the training time and processing power, this model is more efficient.

The model with the highest F1 score for the misinformation classification (Table 5.6) was LLaMa-2, followed by T5 (Causal), 97% and 96%, respectively. These two models have desirable results for this classification task. However, both require high computational power, and LLaMa-2 has an extensive training time (Figure 5.2) compared to BERT, 8.09x more. It is important to notice that in both scenarios, LlaMa-2 and T5 (Causal) outperform the other models by a slight margin. These models contain a Decoder element, which can help them generate and understand text appropriately.

For the BERTScore, we see that both models had an identical performance (Table

5.7). A possible reason is that the RAG process gives sufficient context to generate a coherent response. However, both models have their trade-offs. To use GPT-3.5-turbo, we must pay OpenAI to request their API. On the other hand, LLaMa-3.1 ran with Ollama locally, and we need sufficient memory to execute the model. The use case for each one depends on the user's hardware.

This project focuses on social media posts, and we know that there are frequent changes in how users interact. Additionally, when new diseases are found or named, we must retrain most models to add these words to their vocabulary. Retraining can be costly if the model has many parameters and requires extensive training. Thus, we can say that BERT had overall results that can help combat health misinformation on social media. That model had an F1 score of 88% in health and 92% in misinformation classification. Finally, that model took less overall time to train and used the least amount of RAM.

Chapter 6

Conclusions & Future Work

6.1 Conclusions

In this thesis, we presented how LLMs can be used to refute health misinformation in social media. Additionally, we demonstrated that certain elements within a text—such as mentions, hashtags, and links—play a significant role in shaping its meaning. We also presented how we extracted, processed, stored, and used research papers with LLMs for the misinformation rebuttal. Finally, the research shows that it is possible to fine-tune large models with limited memory using LoRA.

A prototype version of the misinformation rebuttal pipeline was implemented with Python, Postgres, Chroma, and other open-source tools. The research presents the performance results using health-related tweets and misinformation texts from different online sources. Our research preliminary performance results show that we can achieve an F1 score of 90% for health-related classification and 97% for misinformation classification. Additionally, we present that the model can refute misinformation by generating an answer using RAG. The misinformation rebuttal models achieve an F1 BERTScore of 82%. Thus, the project can help health experts combat misinformation and reduce

the risk of negatively impacting public health.

6.2 Future Work

Additional Models for Classification: It is possible to train alternative LLM for the classification tasks.

Processing Time: During our experiment, we noticed that the misinformation pipeline takes excessive time to process. The processing time is mostly caused by the search in the vector database. Exploring different vector databases and similarity search algorithms can improve the processing time.

Rebuttal Helpfulness: The rebuttal the model generates must be as useful as possible for non-technical readers. That rebuttal can include specific information about the disease, such as symptoms, treatment, statistics, and other relevant factors.

Bibliography

[1] S. T and S. Mathew, "The disaster of misinformation: a review of research in social media," *International Journal of Data Science and Analytics*, vol. 13, pp. 1–15, 05 2022.

[2] C. C. Garzón-Alfonso and M. Rodríguez-Martínez, "Twitter health surveillance (ths) system," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1647–1654, 2018.

[3] A. Chernyavskiy, D. Ilvovsky, and P. Nakov, "Transformers: "the end of history" for nlp?," *CoRR*, vol. abs/2105.00813, 2021.

[4] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2024.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[6] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Nee-

lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.

[8] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[10] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[11] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[12] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019.

[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text

transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[14] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[15] J. Y. Koh, R. Salakhutdinov, and D. Fried, "Grounding language models to images for multimodal inputs and outputs," 2023.

[16] X. Deng, V. Bashlovkina, F. Han, S. Baumgartner, and M. Bendersky, "Llms to the moon? reddit market sentiment analysis with large language models," pp. 1014–1019, 04 2023.

[17] Chroma, "The ai-native open-source embedding database," 2022. accessed: 04.27.2024.

[18] pgvector, "pgvector: Open-source vector similarity search for postgres," 2024. accessed: 10.1.2024.

[19] P. N. Singh, S. Talasila, and S. V. Banakar, "Analyzing embedding models for embedding vectors in vector databases," in *2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG)*, pp. 1–7, 2023.

[20] Y. Han, C. Liu, and P. Wang, "A comprehensive survey on vector database: Storage and retrieval technique, challenge," 2023.

[21] T. Sun, A. Somalwar, and H. Chan, "Multimodal retrieval augmented generation evaluation benchmark," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, pp. 1–5, 2024.

[22] F. Shen, *Introduction: Social Media as a News Source*, pp. 1–2. 10 2021.

[23] Z. Epstein, N. Sirlin, A. Arechar, G. Pennycook, and D. Rand, "The social media context interferes with truth discernment," *Science Advances*, vol. 9, no. 9, p. eabo6169, 2023.

[24] M. S. Islam, A.-H. Kamal, A. Kabir, D. Southern, S. Khan, S. Hasan, T. Sarkar, S. Sharmin, S. Das, T. Roy, M. G. D. Harun, A. Chughtai, N. Homaira, and H. Seale, "Covid-19 vaccine rumors and conspiracy theories: The need for cognitive inoculation against misinformation to improve vaccine adherence," 05 2021.

[25] S. Benaissa Pedriza, "Disinformation perception by digital and social audiences: Threat awareness, decision-making and trust in media organizations," *Encyclopedia*, vol. 3, no. 4, pp. 1387–1400, 2023.

[26] T. Yilmaz and Ö. Ulusoy, "Misinformation propagation in online social networks: Game theoretic and reinforcement learning approaches," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 6, pp. 3321–3332, 2023.

[27] A. Harbola, M. Manchanda, and D. Negi, "Misinformation classification using lstm and bert model," in *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, pp. 1073–1077, 2023.

[28] N. Ayoobi, S. Shahriar, and A. Mukherjee, "The looming threat of fake and llm-generated linkedin profiles: Challenges and opportunities for detection and prevention," in *Proceedings of the 34th ACM Conference on Hypertext and Social Media*, HT '23, ACM, Sept. 2023.

[29] D. Villanueva-Vega and M. Rodriguez-Martinez, "Finding similar tweets in health related topics," in *2021 IEEE International Conference on Digital Health (ICDH)*, pp. 184–190, 2021.

[30] "PubMed - National Library of Medicine, url = https://pubmed.ncbi.nlm.nih.gov, note = Accessed: 2024-05-10."

[31] D. C. Comeau, C.-H. Wei, R. Islamaj Doğan, and Z. Lu, "PMC text mining subset in BioC: about three million full-text articles and growing," *Bioinformatics*, vol. 35, pp. 3533–3535, 01 2019.

[32] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-pack: Packaged resources to advance general chinese embedding," 2023.

[33] "Get up and running with large language models., url = https://ollama.com, note = Accessed: 2024-09-29."

[34] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020.

[35] "Hugging Face – The AI community building the future, url = https://huggingface.co, note = Accessed: 2024-10-18."

[36] S. Crone, "Monkeypox misinformation: Twitter dataset," 2022.

[37] Möbius, "Covid-19 fake news dataset," October 2023.

[38] S. Siwakoti, K. Yadav, I. Thange, N. Bariletto, L. Zanotti, A. Ghoneim, and J. N., "Localized misinformation in a global pandemic: Report on covid-19 narratives around the world," March 2021.

[39] Q. Anthony, S. Biderman, and H. Schoelkopf, "Transformer math 101."

[40] "OpenAI, url = https://openai.com, note = Accessed: 2024-09-20."