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

Hi! We encourage you to visit https://libguides.uprm.edu/writingclinics and check out the **Abstracts Clinic.** Keep in mind that depending on your discipline, abstracts should be a **single paragraph**, containing no more than **150 words** for theses or **350 words** for dissertations. It should concisely but clearly summarize your thesis document. The **IMRaD format** is recommended for writing abstracts: Introduction (1-3 sentences long, present tense), Methodology (1-3 sentences long, past tense), Results (1-3 sentences long, past tense), and Discussion (1-2 sentences long, present tense). Remember that the number of sentences and verb tense are only guidelines!

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
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
SQL	Structured Query Language
THS	Twitter Health Surveillance

UPRM University of Puerto Rico Mayagüez

List of Appendices

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 [**?**] 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 [**?**]. 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 [**?**]. 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) [**?**]. 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 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 gather 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 on solving Natural Language Processing problems.

1.3 Contributions

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

- **Pending Contribution Title:** 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 misinformation. 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 to make the correction. We extracted research papers that were added into a vector database. The database helped find similar chunks of texts to use as context for the misinformation rebuttal.

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 medias. For Chapter 3, we can observe the problem description and methodology. Later, on Chapter 4, we have our system architecture, and the projects 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 or 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 computers face when trying to replicate or learn text communication and expressions. However, this changed with the introduction of Large Language Models (LLM) [**?**]. 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 [**?**]. Previous NLP techniques such as Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM) could understand a sentence context in the short term. However, these struggle when trying to understand longer texts. In contrast, transformer architecture, seen in Figure **??**, differs from others because it uses self-attention.

2.1.1 LLM Architectures

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-

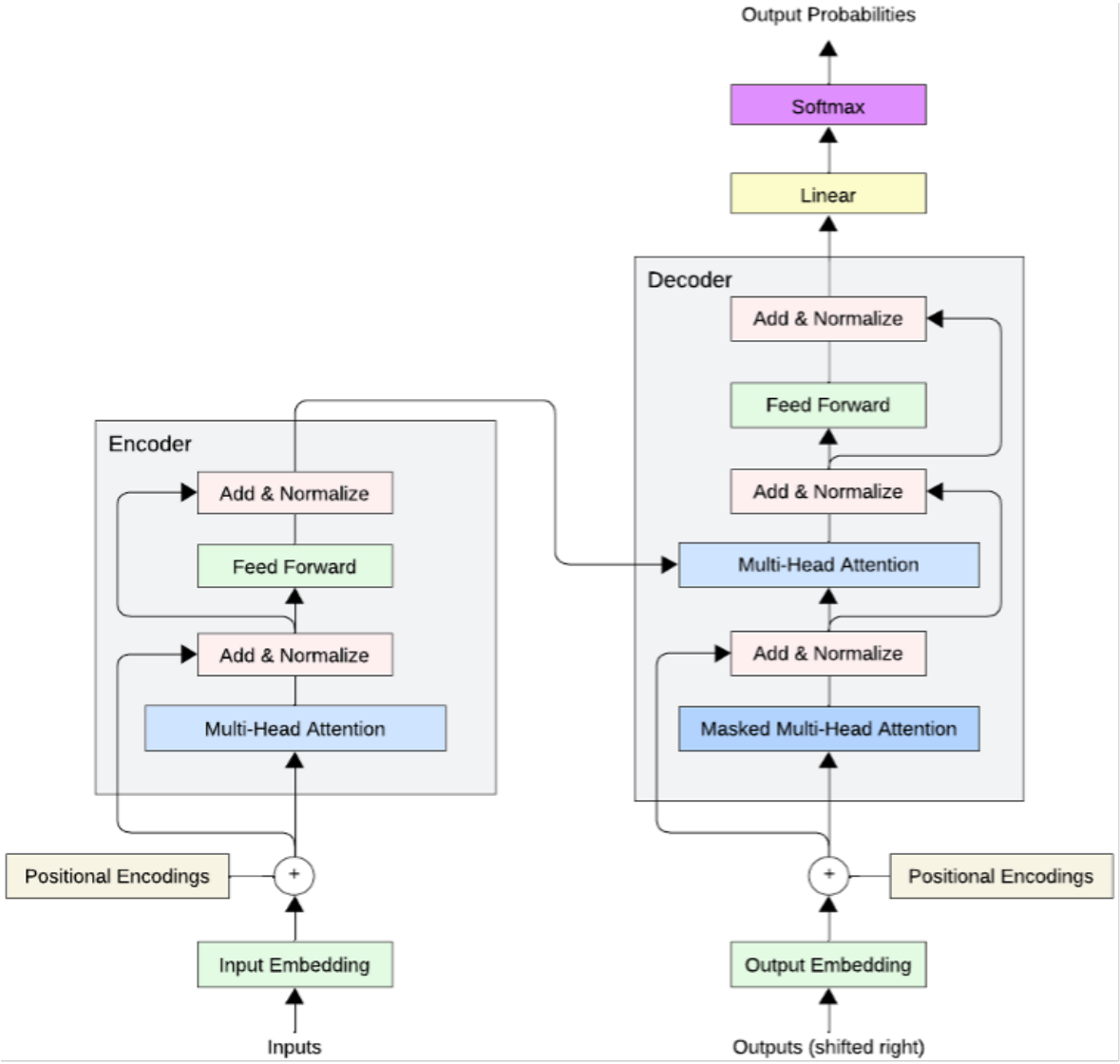

Figure 2.1: The Transformer Architecture

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. In Figure **??**, 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.

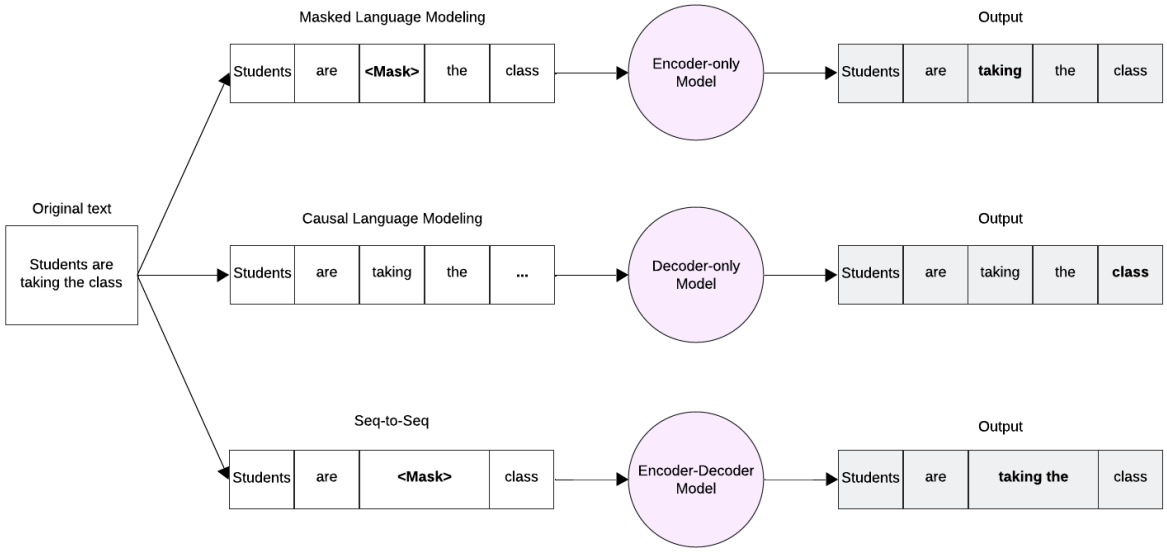

Figure 2.2: LLM Architecture and Comparison

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 **??**, 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 [**?**], Mistral [**?**], and LLaMa [**?**].

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 **??** 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 [**?**] and RoBERTa [**?**].

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 **??**, 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 [**?**] and T5 [**?**] are examples of encoder-decoder LLMs.

2.1.2 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 **??**, 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 finetuning process. Finetuning 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, but now the training data relates to the tasks it will perform.

```
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

2.1.3 Example Applications

The authors in [**?**] 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 captions of those images 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 [**?**], 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 problems 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.

2.1.4 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 [**?**], a native vector database. Also, some relational databases have extensions that allow them to make vector similarity searches, like Postgres with pgvector [**?**]. They can turn images, videos, documents, and others into numerical embedding. These embeddings are numerical vectors capturing semantic meaning [**?**]. 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 **??**. 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.

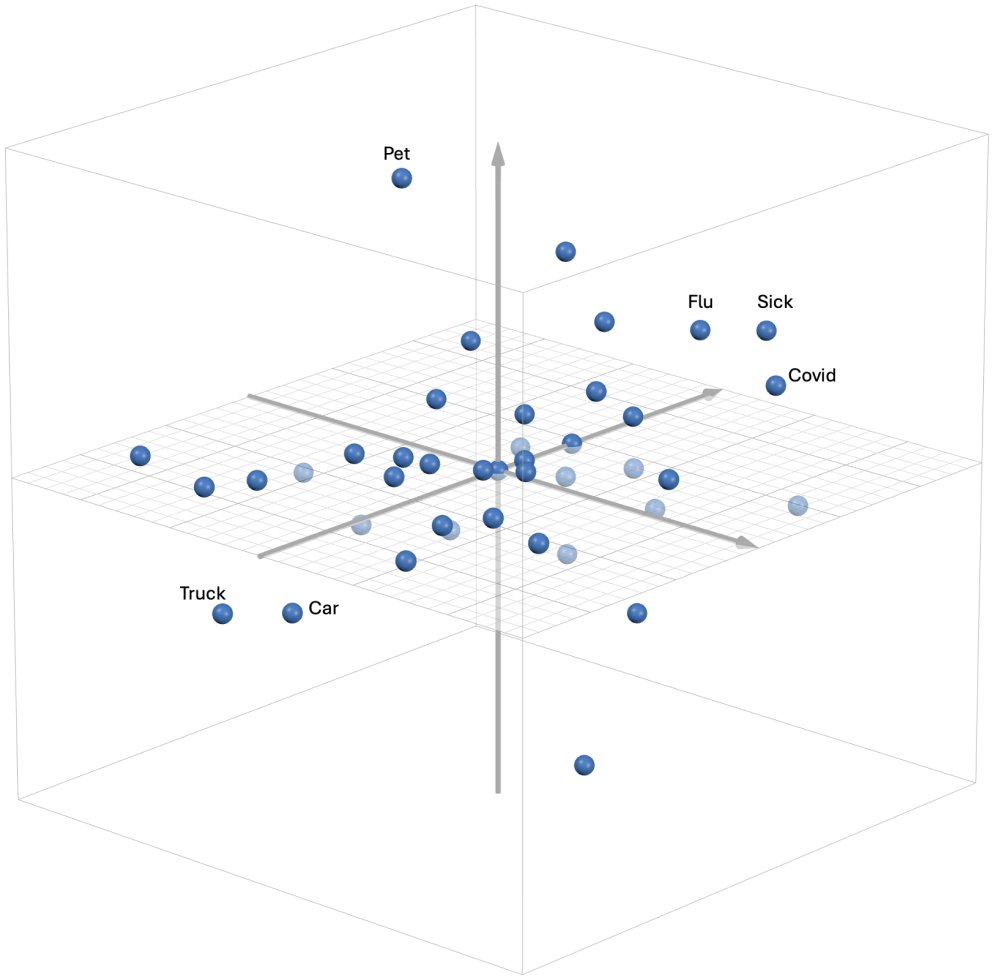

Figure 2.4: High-Dimensional Space Representation

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 [**?**]. 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 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 [**?**] 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 [**?**] and occasionally take this information as truth without validation [**?**]. 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 [**?**], which led to high health risks or even deaths [**?**] 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 [**?**].

There have been various research studies on reducing the propagation of misinformation. Misinformation was modeled as a game-theoretic problem in [**?**], 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 [**?**] 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 [**?**], 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 [**?**]. 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 **??**. 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%.

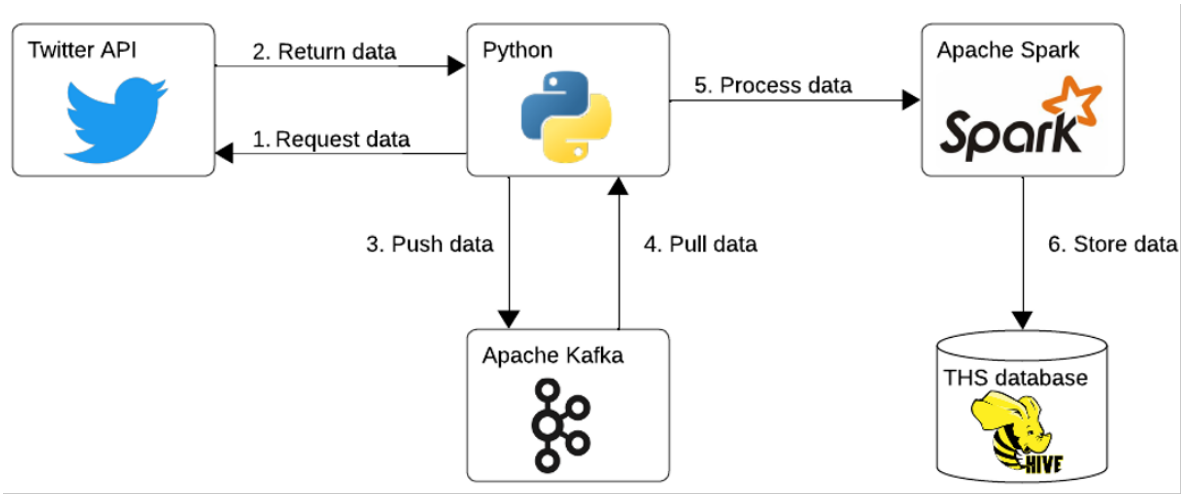

Figure 2.5: THS Architecture

Later the project was updated to find similarities between tweets [**?**]. 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 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.

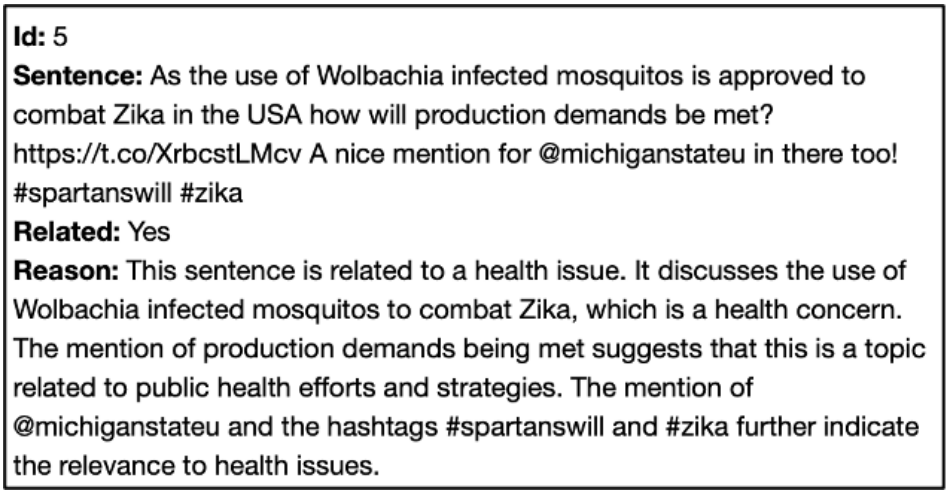
Id: 5
Sentence: As the use of Wolbachia infected mosquitos is approved to combat Zika in the USA how will production demands be met? https://t.co/XrbcstLMcv A nice mention for @michiganstateu in there too! #spartanswill #zika
Related: Yes
Reason: This sentence is related to a health issue. It discusses 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

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 **??**, 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

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

Chapter 3

Methodology

3.1 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.

3.2 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 **??**.

Table 3.1: Cluster's Node Specifications

Hardware	Description
Hard Disk	256GB
RAM	80GB
Processor	#
GPU	NVIDIA V100 32GB VRAM

3.3 Software

There were various software tools used for the project. The node used for the experiment run Ubuntu Linux 20.04 LTS. 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) [**REFERENCE**]. We used BERT, T5, and LLaMa2 for our research. In Table **??** we see the specifications for each model. Also, we use Postgres 14 to store our extracted research papers. In addition, we implemented Chroma 0.4.24 as our vector database. Finally, for the RAG process, we installed Ollama tu run LlaMa3.1 8B parameter model.

Table 3.2: 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

3.4 Datasets

This sections explains the two datasets used for the training process.

3.4.1 Health-related Dataset

The health-related dataset comprises over 12,441 tweets extracted from the previous THS project. Health professionals labeled the dataset into three categories: related (7848), unrelated (3828), and ambiguous (765) tweets. However, this dataset is imbalanced; thus, we applied class weights to the loss function. These weights give a higher penalty for classes that appear frequently. The weights prevent the models from overfitting, mitigating the effects of class imbalance during training.

Before we start finetuning the models, we preprocessed the data. 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. 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.

3.4.2 Misinformation Dataset

The misinformation dataset, with 8,749 texts, combines data from different sources such as news, social media, and blogs. These records were labeled as misinformation (3638) and non-misinformation (5111) ***[REFERENCE]***. We also applied class weights for the data imbalance. The dataset in [**?**] consists of social media texts that mentions the monkeypox. We only used the "text" and "binary_class" columns from their dataset.

Another source was [**?**], they classified Covid-19 texts. That dataset contains articles and post 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 assign to the file. The reason for this is that, occasionally, online users share news reports or articles based on the headline alone. Next, the last dataset [**?**]

3.5 Fine-tuning

The Large Language Models (LLM) 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. This process is called fine-tuning, we modify an existing model to accomplish a task. In this case, we taught the models to classify two types of texts: health-related and misinformation-related texts.

Three models were identified to perform these classification tasks. The models used for this experiment are Bert, T5, and LLaMa-2. Each one has a different architecture.

3.5.1 Health-Related Classification

3.5.2 Misinformation Classification

2. Misinformation classification

LR, Batch size, seed, and epochs are static.

- Each model was fine-tune twice.

1. Sequence classification: 1, 2, or 3; 1 or 2.

2. Classification with text generation: Related, Unrelated, or Ambiguous; Misinformation or Not Misinformation.

- Weighted average added for the sequence classification.

Chapter 4

System Architecture

4.1 Paper ETL Pipeline

Our model must use credible sources of information to rebut misinformation. We identified PubMed [**REFERENCES**], 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 [**?**], 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 **??** shows the processes of data extraction.

4.1.1 Scraper

The first step of the pipeline was identifying what papers we needed to extract. We selected 17 topics for the data extraction process. These topics are:

- allergy
- cancer
- bird flu
- chickenpox

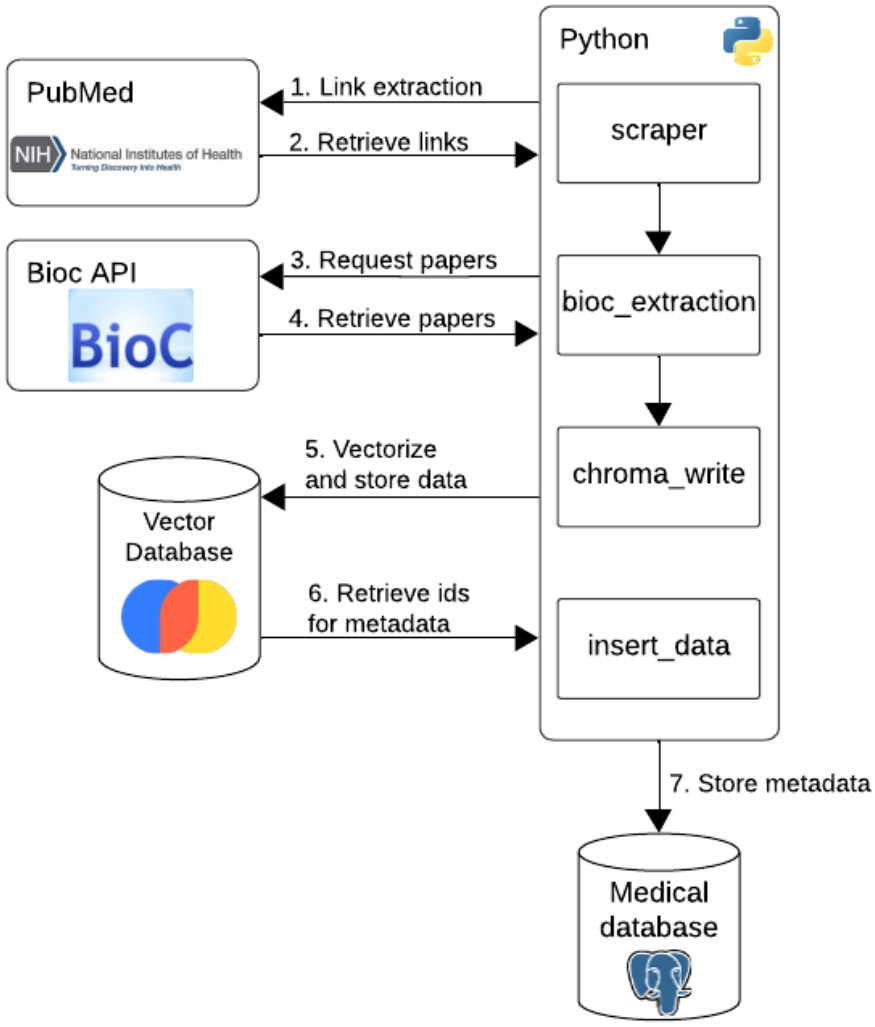

Figure 4.1: Medical Data Extraction Pipeline

- common cold
- conjunctivitis
- covid sickness
- covid symptoms
- covid treatment
- covid vaccine
- flu vaccine
- headache
- influenza
- monkeypox
- stomach aches
- swine flu
- zika

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 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.

4.1.2 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.

4.1.3 Vectorizing data

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

4.1.4 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 **??**. 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.

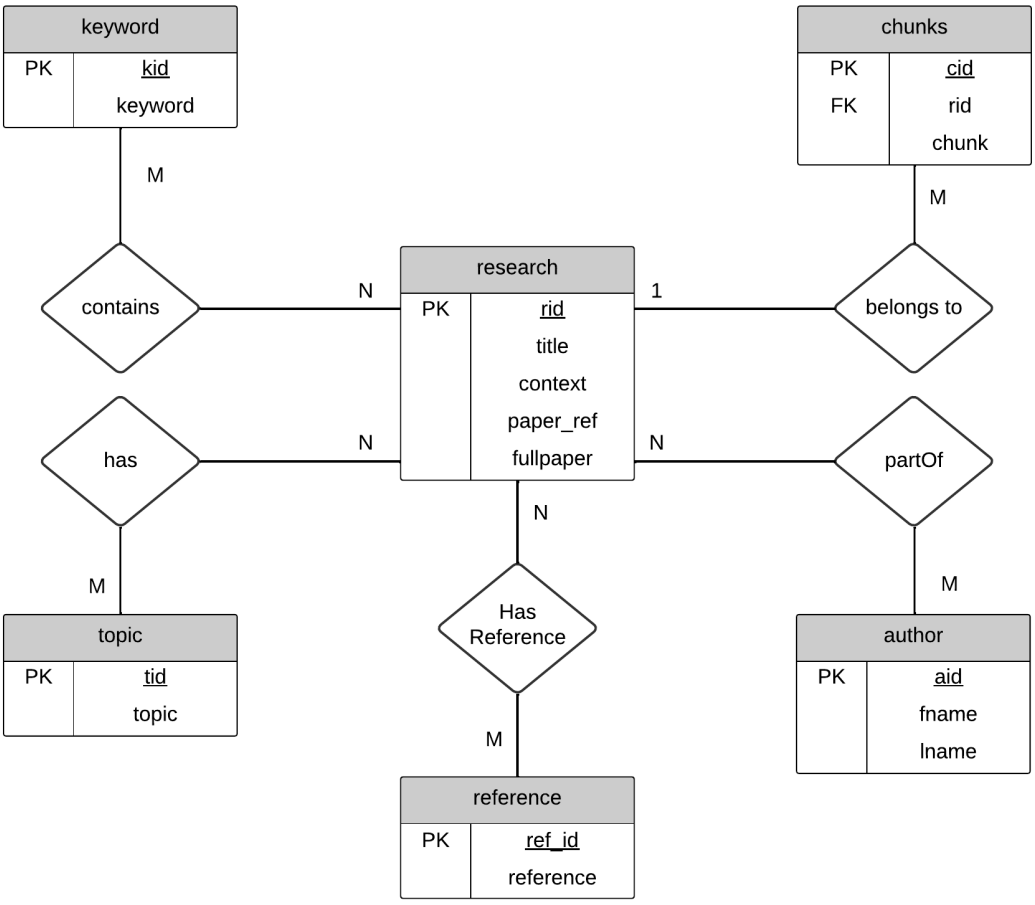

Figure 4.2: Research Papers Schema Diagram

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.

4.2 Misinformation Rebuttal Pipeline

After training the models and storing the context for the rebuttal, we create the model pipeline. The pipeline shown in Figure **??** shows the process of receiving a text, making the classifications, and returning an explanation of why it is misinformation.

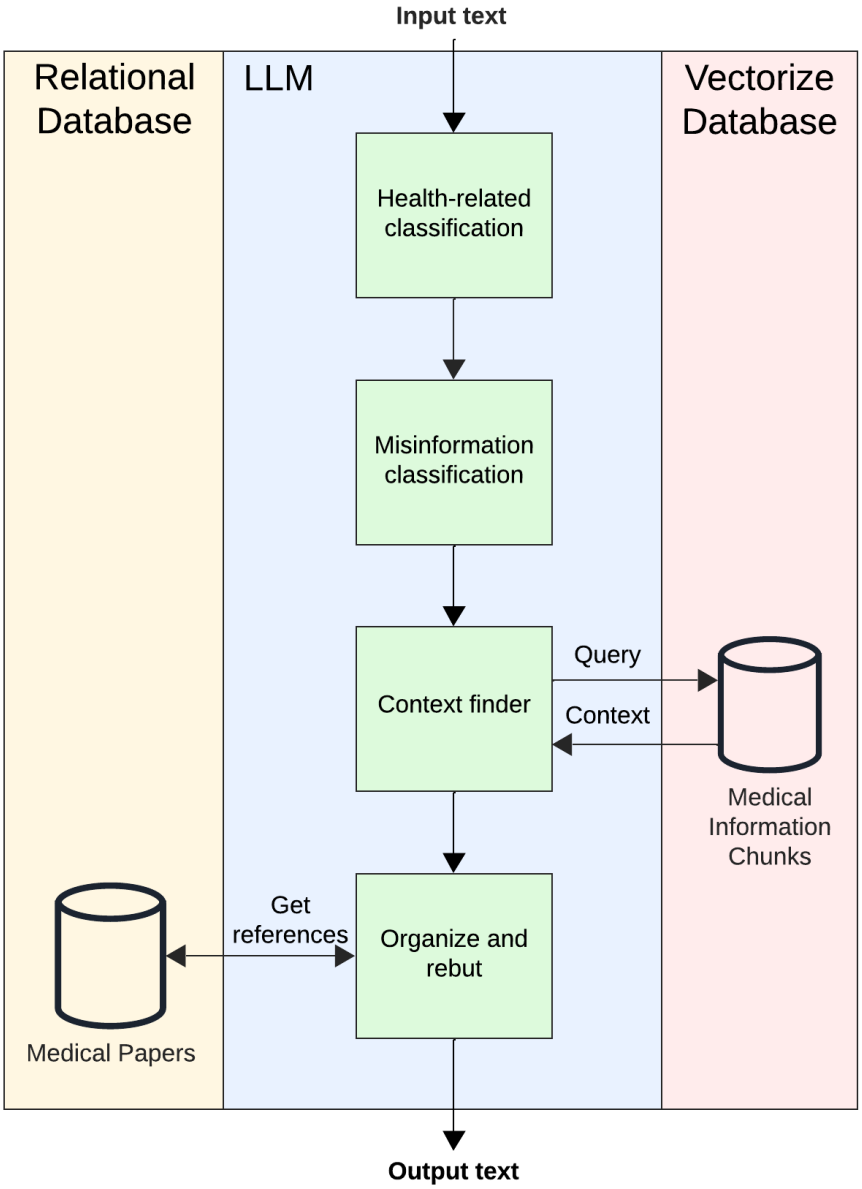

Figure 4.3: Misinformation Rebuttal LLM System Architecture

4.2.1 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.

4.2.2 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.

4.2.3 Context Finder

Before starting the search on the vector database, we must understand the topic of the text. To understand the text, 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. These chunks are then sent to another model to be analyzed and organized.

4.2.4 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.

Chapter 5

Performance Evaluation

5.1 Hardware

- V100 machines: 32Gb VRAM, and 80-ish RAM

  Cuda 11.7

- Python 3.9.19

- Pytorch 2.0.1

- Transformers 4.34.0

5.2 Software

Various softwares were used for this research, these are:

- Python
- Hugging Face
- BioC

- Torch
- CUDA
- Transformers

Chapter 6

Conclusions & Future Work

6.1 Conclusions

6.2 Future Work

- Model Helpfulness:

- Memory Optimization:

-

Appendix A: MATLAB Code

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis

ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Appendix B: Data

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Appendix C: More Data

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Appendix D: More Data

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.