FIGHTING HEALTH-RELATED MISINFORMATION IN SOCIAL MEDIA WITH LARGE LANGUAGE MODELS

Moisés Robles Pagán
Electrical and Computer Engineering Department
University of Puerto Rico - Mayagüez
moises.robles@upr.edu

Manuel Rodríguez Martínez
Electrical and Computer Engineering Department
University of Puerto Rico - Mayagüez
manuel.rodriguez7@upr.edu

***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 an investigation that can help health experts combat and rebut disinformation on different social media platforms.**

***Index Terms*—Large Language Model, Misinformation, Transformers, Vector Databases**

I. Introduction

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 users 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. 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 [3–5]. Instead of using the THS architectures, we opted for Large Language Models (LLM).

We employ an LLM in this investigation, to 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 when hashtags, mentions, and emojis are removed. 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 with an F1 BertScore of 82%. Hence, our system proved that it is possible to classify and rebut health-related misinformation.

A. Contributions

This paper provides the following original 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.
- **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. 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.

B. Paper Organization

This paper has the following organization. Section II contains the background on the transformer and the Large Language Models architectures, vector databases, and the Twitter Health Surveillance (THS). For section III, we can observe the system architecture for the data extraction and classification process. Later, in section IV we show the system performance. Related works are presented in section VI. Ending with section VII, we have our conclusion with suggestions for future work.

II. Background

A. Large Language Models (LLM)

The are three different architectures for Large Language Models, encoder-only, decoder-only, and encoder-decoder. Each one has advantages on specific tasks. In Figure 1

1) Encoder-only models: These models are like BERT. This type of model predict by masking specific words in a sentence. They are better for classification and sentiment analysis.

2) Decoder-only models: For these model we have the well known GPT-3. They receive one input and try to predict the entire text. They are good for summarizing and text-generation.

3) Encoder-Decoder models: T5 is an encoder-decoder models. They mask entire sequences of text. Good for translation and question and answering.

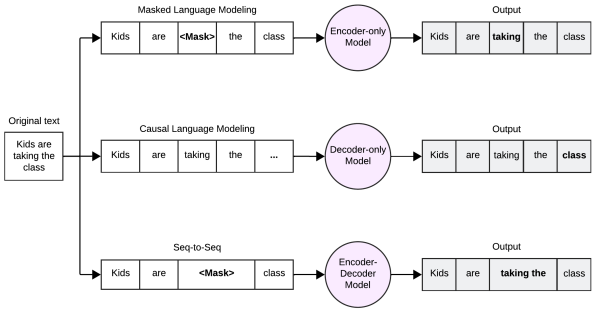

Fig. 1: The transformer architecture

III. System Architectures

This section presents the general architectures of the systems we used for the development and experiments in our research. We illustrate the Extraction, Transformation, and Load (ETL) pipeline we used for the medical research papers. Additionally, we have our misinformation classification process, where we classify the health misinformation and refute it. Finally, we have the system's User Interface (UI).

A. *Research Paper ETL Pipeline*

Our model must use credible sources of information to rebut misinformation. We identified PubMed [6], 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 [7], 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 2 shows the processes of data extraction.

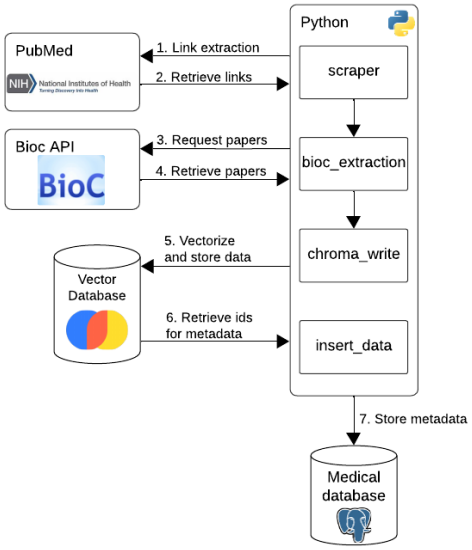

Fig. 2: Medical Data Extraction Pipeline

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 I.

TABLE I: 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	

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.

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 extract 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) Vectorizing data: After retrieving the data, we need to vectorize the papers. Figure 3 shows the process of the papers vectorization. First, each research paper's context was split into chunks using LangChain. Then, we used an LLM, BAAI [8], to embed these chunks. A universal unique identifier (UUID) was combined with each chunk and stored in a Chroma [9] database. After uploading the data to Chroma, we added these UUIDs to their JSON.

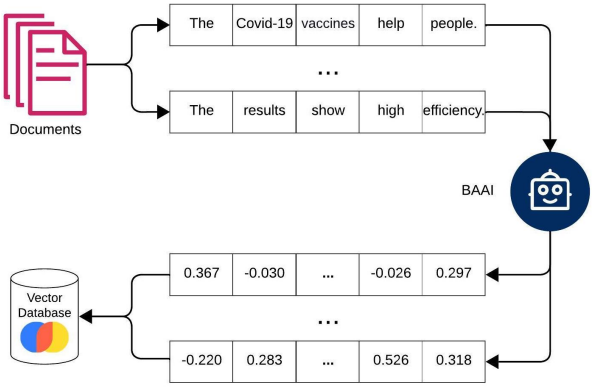

Fig. 3: Data Vectorization Process

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 4. The tables in this schema are as follow:

Research: Contains the research paper's metadata. Its attributes are: title, paper's title; context, the paper's text; paper_ref, the reference of the paper; and fullpaper, a boolean that is true if the paper contains an abstract, introduction, methodology, discussion, conclusion, and references.

Chunks: Pairs the UUIDs from the paper's chunks and their respective research record.

Keyword: 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: The topics used to search the papers.

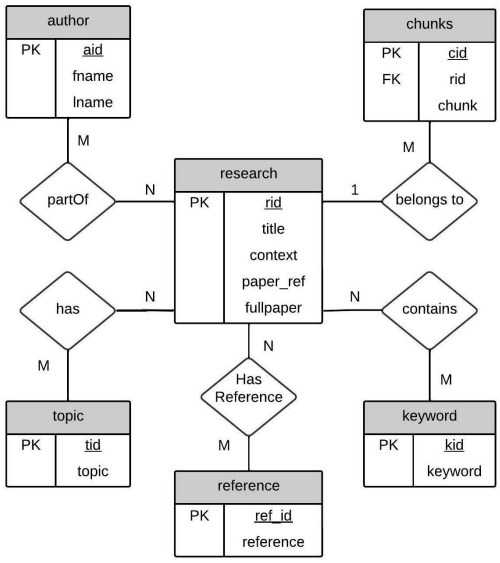

Fig. 4: Research Papers Schema Diagram

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

B. Misinformation Rebuttal Pipeline

After training the models and storing the context for the rebuttal, we create the model pipeline. The pipeline shown in Figure 5 shows the process of receiving a text, making the classifications, and returning an explanation of why it is misinformation.

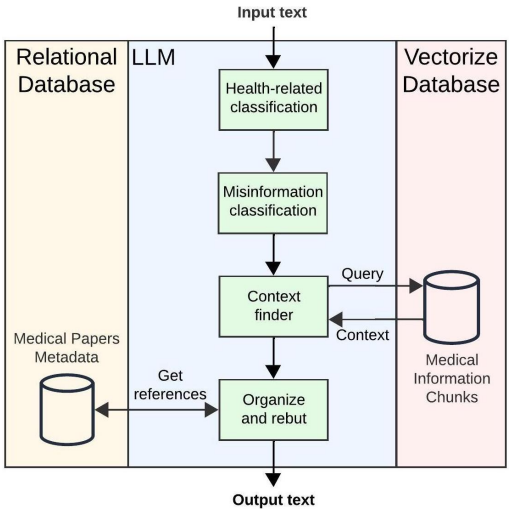

Fig. 5: Misinformation Rebuttal Pipeline

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 fine-tuned to classify health-related texts. This model determines whether the input is related, unrelated, or ambiguous. If the text is health-related, we go to the next part of the pipeline. When the text is non-related or ambiguous, we end the process because it is out of the model's scope.

2) Misinformation Classification: After confirming that the text is health-related, we validate that it contains misinformation. The fine-tuned 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 that classification. When the result contains misinformation, we start the search for research papers to be used to rebut the message.

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 [10]. 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 & @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.

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 so the model can evaluate and generate an explanation that rebuts the misinformation. Ollama returns a 2-3 sentence result that explains the text's misinformation. The final output is a JSON with the classifications, rebuttal, and references used for the rebuttal.

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.

IV. Performance Evaluation

A. System Setup

1) Performance Metrics: To evaluate the models effectiveness we used the following metrics:

- **Precision**: To evaluate the proportion of positives examples that the model classified as positive that are actually positive.
- **Recall**: To evaluate the proportion of the positives examples that the model classified correctly against all positives.
- **F1**: To balance the precision and recall scores.
- **Elapsed Time**: The time needed to complete the models finetuning stage.
- **BERTScore**: Compute a similarity score between two different sentences, to verify that the rebuttal generated by the LLM relates to the classified text [11].

2) Hardware: The node's specification can be found on Table II.

TABLE II: Cluster's Node Specifications

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

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 CUDA 11.7. To initialize and use the models we relied on the Transformers 4.34.0 library. The models were imported from Hugging Face (HF) [12]. 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 III we see each model specification. Also, we use Postgres 14 to store our extracted research papers. In addition, we added Chroma 0.4.24 as our vector database. Next, for the RAG process, we installed Ollama [10] and LangChain 0.1.16 to run LlaMa3.1 (8B) [13] parameter model. Finally, to create the UI to show the results we used React.

TABLE III: 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

B. Fine-tuning

The LLMs used for this paper are pre-trained base models, 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 fine-tuned the models to achieved our classification goals. 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 III. 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. Because of the resource limitations, we fine-tuned the models using LoRA [14].

1) Parameter-Efficient Fine-Tuning (PEFT): PEFT is a library that reduces the memory usage of a model to be fine-tuned. That library allows us to use the Low-Rank Adapter (LoRA). That adapter helps us reduce a model's trainable parameters. The LoRA hyperparameters used are found on Table IV.

TABLE IV: LoRA Hyperparameters

Parameters	Value
r	16
alpha	32
dropout	0.05
bias	all

2) 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 V, and the description for each are as follows::

TABLE V: 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

C. Health-Related Classification Results

We present the results of the health classification process and compare them with the best overall model of the previous THS project [2]. That work concluded that their best model is an LSTM, with no attention, and a GRU layer.

TABLE VI: Health Related Precision Result

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

1) Precision: Table VI 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 score of 48%.

TABLE VII: Health Related Recall Result

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

2) Recall: Table VII 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% [2]. In contrast, most of our models had a score of at least 80%. Here, our best model was T5 (Causal), with a performance of 95% in recall. Our model with the highest precision, LLaMa-2, ended with 84%.

TABLE VIII: Health Related F1 Result

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

3) F1: Table VIII 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.

4) Training Time: In Figure 6, we present our model's training time. BERT trained faster than any other model, which took 3.12 hours. The T5 (Sequence) model took 24.03 hours, while the T5 (Causal) model finished in 26.15 hours. Lastly, LLaMa-2 took 85.43 hours, over three days, to train. Based on these results, we can infer that models with fewer parameters train faster. BERT trained 27.4x faster than LLaMa-2.

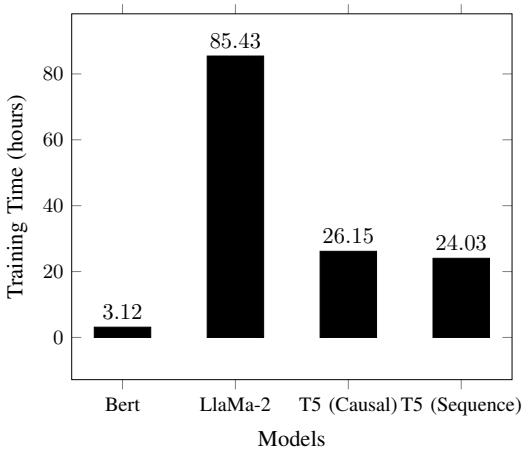

Fig. 6: Health-Related Models Training Time

D. Misinformation Classification Results

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.

TABLE IX: Misinformation Precision Result

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

1) Precision: Table IX 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%.

TABLE X: Misinformation Recall Result

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

2) Recall: Table X shows the recall metric's results for the misinformation classification. When we compare these results with the precision table (Table IX), 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.

TABLE XI: Misinformation F1 Result

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

3) F1: Table XI 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%.

4) Training Time: In Figure 7, 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 6), 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 1.54 hours to train. Next is T5 (Causal) with 6.18 hours and T5 (Sequence) with 3.48 hours. Finally, LLaMa-2 took 47.33 hours 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.

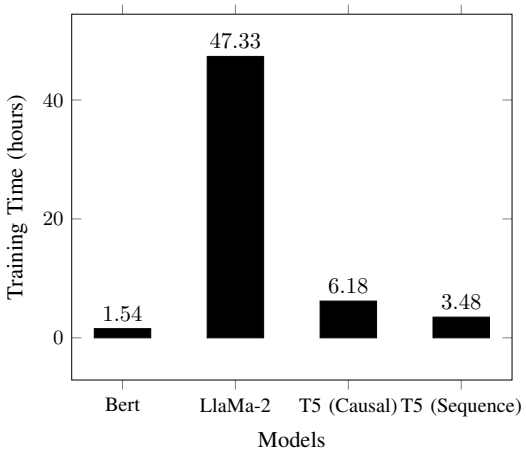

Fig. 7: Misinformation Models Training Time

E. BERTScore Result

Our model BERTScores' F1 is shown in Table XII. 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 XII: BERTScore F1 Results

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

F. 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 VIII) 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 6) 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 XI) 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 7) compared to BERT, 30.90x 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 XII). 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.

V. Related Work

Lorem ipsum

VI. Conclusion

In this paper, 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. Our system 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 system can help health experts combat misinformation and reduce the risk of negatively impacting public health.

VII. Acknowledgment

This research is supported by the United States (US) National Library of Medicine of the National Institutes of Health (NIH) under award number R15LM012275. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References

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

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

[5] 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.

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

[7] 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.

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

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

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

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

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

[13] 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.

[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.