Harnessing RAG for Harry Potter Question Answering

Mitesh Jalan
Stony Brook University
msjalan@cs.stonybrook.edu

1 INTRODUCTION

We introduce RAGwarts, a RAG[1] based LLM for answering questions based on the Harry Potter universe. An interactive interface to answer your questions about your favorite fiction universe is the most innovative way to bring it to life. This project, a question-answering model tailored for Harry Potter enthusiasts, holds significant importance for several reasons. Firstly, it caters directly to a fanbase deeply invested in the intricacies and mythology of the Harry Potter universe. By furnishing accurate and insightful responses to their queries, we can enrich their engagement with the franchise, fostering a stronger sense of connection and satisfaction among fans.

In terms of NLP coverage, This project spans various sub-fields including question-answering, language modeling, and information retrieval. Through fine-tuning multiple small language models on Harry Potter trivia questions, we tackle the challenge of understanding domain-specific language and context. Additionally, the incorporation of RAG for generating responses represents an innovative approach to merging retrieval-based and generative methods, showcasing the interdisciplinary nature of NLP research and development.

One technical challenge we aim to tackle here is the effective integration of multiple models within the RAG framework to provide coherent and accurate responses. We also try Instruction Tuning the models on trivia questions to make them better at answering questions the fans might ask. RAG along with Instruction tuning of the models provide significant advantage over zero-shot question answering. We take multiple small language models with parameters ranging from 1-3 billion and compare their performance over answering trivia questions based on Harry Potter using RAG with and without Instruction fine-tuning them. We present a comprehensive review of how the models perform using various statistical evaluation scores along with human evaluation.

2 BACKGROUND

Since pre-trained LLMs lack the specialized knowledge and context required to accurately answer questions about the intricate details, characters, and events within the Harry Potter universe, we employed several recent ideas and methods in NLP to tackle this problem and equip the models with methods that will help tackle specific questions without finetuning the entire model which is extremely difficult for LLMs. We explore several key innovations, including the Retrieval-Augmented Generation (RAG) framework, instruction tuning, the use of small language models, and techniques like LoRA[2] for fine-tuning large models.

The RAG framework combines the strengths of retrieval-based and generative models. RAG utilizes a retriever to select relevant passages from a large corpus of text and a generator to produce coherent responses based on the retrieved information. This approach allows for more nuanced and contextually rich responses. Instruction tuning is a technique used to fine-tune language models for specific tasks or domains by providing explicit instructions or examples during the training process. This method enables models to learn task-specific behaviors and understand domain-specific language and context more effectively. We use small language models which are the compact versions of larger pre-trained models having 1-3 billion parameters. These models offer a more lightweight and resource-efficient alternative while still retaining significant language understanding and generation capabilities. We use small language models as we have limited computational resources and need to focus on real-time performance. LoRA (Low Rank Adaption) is a technique used for fine-tuning large language models to improve their performance on specific tasks. By adjusting the relevance of different layers in the model architecture during fine-tuning, LoRA allows for more targeted adaptation to the task at hand. This approach helps prevent catastrophic forgetting and enables large models to adapt more effectively to new tasks or domains.

2.1 Tokenization using LLM Tokenization for RAG Systems:

Given a document D, the text is split into tokens $t_1, t_2, \ldots, t_n$ using the tokenizer associated with the pre-trained LLM. This tokenizer applies subword segmentation methods such as Byte Pair Encoding (BPE), WordPiece, or SentencePiece.

The document is represented as a sequence of characters $c_1, c_2, \ldots, c_m$. The tokenizer applies a mapping:

$$T(D) = \{t_1, t_2, \ldots, t_n\}, \quad t_i \in V$$

where V is the tokenizer's vocabulary. The number of tokens n depends on the granularity of tokenization:

$$n \leq m \quad \text{(subword tokens reduce length).}$$

For example: - "hello" $\rightarrow$ ["hel", "lo"] (SentencePiece).

2.2 Embedding Tokens:

Each token t_i from the tokenizer is mapped to an embedding $\mathbf{e}_i \in \mathbb{R}^d$ using the LLM's embedding matrix $\mathbf{E}$:

$$\mathbf{e}_i = \mathbf{E}[t_i]$$

Here, $\mathbf{E} \in \mathbb{R}^{|V| \times d}$ is the embedding matrix of the LLM, pre-trained and shared across all uses of the model.

For a document D of n tokens, the embedding sequence is:

$$\mathbf{E}(D) = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}, \quad \mathbf{e}_i \in \mathbb{R}^d$$

3 DATA

We needed data for two tasks, one for storing in the vector database for the retriever part of RAG to be able to do a query match and fetch the top k matches. These matches are then fed into the prompt along with the user question to allow the model to have context around the question and answer it in a better way. The second task is fine-tuning as well as instruction tuning the small language models. We do this to allow models to learn about the Harry Potter Universe and answer intricate questions.

For the first task of creating a vector database, we used the textual data of the Harry Potter novels, and screen plays written by J.K. Rowling. These novels and screen plays have a combined total of around 1.7 million words which come to around 1.8 million tokens after tokenization using the respective model's tokenizer. We sourced this data[3] from the internet in the form of pdfs. We also decided to use the transcripts of the Harry Potter movies which have around 250k tokens in the vector-store to provide more context to the model. We used DirectoryLoader from LangChain to load the pdfs into a FAISS [4] datastore using the RecursiveCharacterTextSplitter to tokenize the sentences of the pdfs.

For the second task, which is instruction tuning, we used a HuggingFace dataset of Harry Potter trivia questions[4] consisting of question and answer-columns. The dataset has more single-word answers with direct questions while some of the answers are more descriptive in nature. There are around 3k question-answer pairs in the train set and 500 pairs in the test set of the dataset. We used the train set for instruction tuning the models. For each model, we had to use a different format for instruction tuning. Llama used a special [INST] token to contain the question while phi-2 used Human assistant format. We used the test set of the dataset for evaluation of the models using statistical scores like BLEU [5], METEOR [6], Cosine Similarity, and ROUGE [7].

4 METHODS

4.1 Retriever

In the retriever component of this RAG model for the Harry Potter universe, we employed several key techniques and tools to enhance the retrieval process and improve the relevance of retrieved passages. Central to this approach was the utilization of a FAISS (Facebook AI Similarity Search) vector store to efficiently index and store embeddings of the Harry Potter dataset. To extract text from the Harry Potter dataset, we employed LangChain, a versatile library for loading PDF documents. This allowed us to seamlessly access and process the textual content of the dataset for indexing. Additionally, we utilized the RecursiveCharacterTextSplitter to split the text into tokens, ensuring granularity and accuracy in the indexing process.

Once the text was tokenized, we employed the Sentence transformers [8] model to generate embeddings for each passage. This model, based on transformer architecture, is specifically designed to capture semantic similarities between sentences and paragraphs. By creating embeddings for the text passages, we transformed the textual data into high-dimensional vectors, enabling efficient similarity search and retrieval using FAISS.

4.1.1 *Vector Similarity Search.*

Corpus Representation: Let the corpus of documents be represented as:

$$C = \{d_1, d_2, \ldots, d_N\},$$

where d_i is the i-th document.

Tokenization and Embedding: Each document d_i is tokenized into $\{t_1, t_2, \ldots, t_m\}$, and embeddings are computed using a function $f : d_i \to \mathbb{R}^d$, where d is the embedding dimension.

The document embedding is given by:

$$v_i = \text{Mean}\left(\{f(t_1), f(t_2), \ldots, f(t_m)\}\right) \in \mathbb{R}^d.$$

Similarity Calculation: The similarity between a query vector $q \in \mathbb{R}^d$ and a document vector v_i is computed as:

$$S(q, v_i) = \cos(q, v_i) = \frac{q \cdot v_i}{\|q\| \cdot \|v_i\|}.$$

Top-k Selection: The top-k documents with the highest similarity scores are retrieved:

$$\text{Retrieve}(q, C) = \{d_{i_1}, d_{i_2}, \ldots, d_{i_k}\}, \quad i_j \in \arg\max_i S(q, v_i).$$

To reduce complexity, FAISS applies vector quantization:

$$J = \sum_{i=1}^{N} \min_{c_j \in C} \|v_i - c_j\|^2,$$

where $C = \{c_1, c_2, \ldots, c_k\}$ are the cluster centroids obtained using k-means clustering.

4.1.2 *FAISS: Nearest Neighbor Search.*

FAISS employs k-nearest neighbors (k-NN) in high-dimensional space. **Vector Quantization** To reduce complexity, FAISS applies vector quantization:

$$J = \sum_{i=1}^{N} \min_{c_j \in C} \|v_i - c_j\|^2,$$

where $C = \{c_1, c_2, \ldots, c_k\}$ are the cluster centroids obtained using k-means clustering.

Approximate Search Given a query q, the search complexity is reduced by comparing q only to the centroids c_j:

$$\hat{i} = \arg\min_j \|q - c_j\|.$$

4.2 Generator

In the generator component of this RAG (Retrieval-Augmented Generation) model for Harry Potter trivia, we employed small language models such as TinyLlama and Phi-2 (developed by Microsoft) and available on the HuggingFace platform. These auto-regressive models excelled at generating coherent and contextually relevant text, making them ideal candidates for this text generation needs.

To streamline the text generation process, we created a HuggingFace pipeline specifically for text generation, configuring it with parameters such as the tokenizer and temperature to control the diversity and fluency of the generated text. By encapsulating the text generation logic into a pipeline, we ensured consistency and efficiency in this generation workflow.In addition to the generator, we integrated a RetrieverQA framework built upon LangChain, library for document processing. Leveraging the vector database created earlier, which stored embeddings of the Harry Potter dataset, the RetrieverQA framework enabled us to efficiently retrieve relevant passages based on user queries.

To generate responses, we created a prompt template that included the user's question and the context retrieved from the vector database. This template served as input to the generator, providing the necessary information for generating contextually appropriate answers. By incorporating the user's question and retrieved context into the prompt, we ensured that the generated responses were tailored to the specific query and context provided.Within the RetrieverQA framework, we configured the retriever chain with parameters such as chain type ('stuff'), number of retrieved passages (k=3), and search type ('similarity'). These parameters optimized the retrieval process, ensuring that the retrieved passages were highly relevant to the user's query.

Once the relevant passages were retrieved, we invoked the QA chain with the user's question and processed the generated answer, including formatting and citation of the sources used from the Harry Potter text. This ensured that the generated responses were not only accurate but also properly attributed to the original sources within the Harry Potter dataset.

4.2.1 ***Generative Component***.

Given the retrieved documents $D = \{d_{i_1}, \ldots, d_{i_k}\}$ and the query q, the generator creates a response r by maximizing the conditional probability:

$$r = \arg\max_{r'} P(r' \mid q, D; \theta),$$

where θ represents the parameters of the trained language model.

The output probability is computed as:

$$P(r' \mid q, D; \theta) = \prod_{t=1}^{T} P(r'_t \mid r'_1, r'_2, \ldots, r'_{t-1}, q, D; \theta),$$

where T is the length of the generated response.

4.3 Instruction tuning

In the Instruction Tuning phase, we fine-tuned the models on a dataset consisting of question-answer pairs based on the Harry Potter universe. Given the challenges associated with fine-tuning large models, we adopted LoRA (Low-Rank Adaption) to facilitate the tuning process. LoRA allowed us to adapt the models to the specific task of question-answering effectively.

Utilizing the Trainer API, we trained the models for 5 epochs while monitoring the training loss. To prepare the input instances for training, we employed the AutoTokenizer provided by the model and used its padding token for padding. Additionally, we implemented Parameter Efficient Fine Tuning (Peft) [9] techniques to optimize the fine-tuning process. LoRA is used to adjust the relevance of different layers in the model architecture during fine-tuning. Rather than updating all layers uniformly, LoRA allows for layer-specific adjustments based on their importance for the target task. This enables the model to preserve its ability to perform well on previous tasks while adapting more effectively to new tasks.

We began by configuring a LoRAConfig, setting parameters such as r and lora alpha. Experimenting with different values for these parameters, we aimed to find the optimal settings for fine-tuning. To expedite the training process, we employed techniques like gradient accumulation and batching, enhancing the efficiency of model training. Each fine-tuning session lasted approximately two hours for 5 epochs for each model. We utilized the Adam optimizer for parameter updates during training. Specifically, we employed the Supervised Finetuning Trainer class to oversee the fine-tuning process. Adapting the input format to suit the instruction tuning task, we structured the input differently for each model. For TinyLlama, we used the format <s> [INST] <question> [/INST] answer </s>, while for Phi-2, we utilized the format Human: <question> Assistant: <answer>. This ensured that the models received input tailored to their specific instruction tuning requirements, facilitating effective adaptation to the task of question-answering in the Harry Potter context.

4.3.1 ***Mathematical Intution***.

For a given input-output pair (x, y), the goal is to minimize the following loss function:

$$L = -\sum_{(x,y)\in\mathcal{D}} \log P(y \mid x; \theta),$$

Where:

- $\mathcal{D}$ is the instruction-tuning dataset, consisting of pairs (x, y), where x is an instruction and y is the desired output.

- $P(y \mid x; \theta)$ is the conditional probability that the model assigns to output y given input x, parameterized by θ.

- θ represents the model parameters (e.g., weights of the neural network).

- The summation is over all examples in the dataset.

This loss function encourages the model to predict outputs y that maximize the probability assigned to the true output, given the input instruction x.

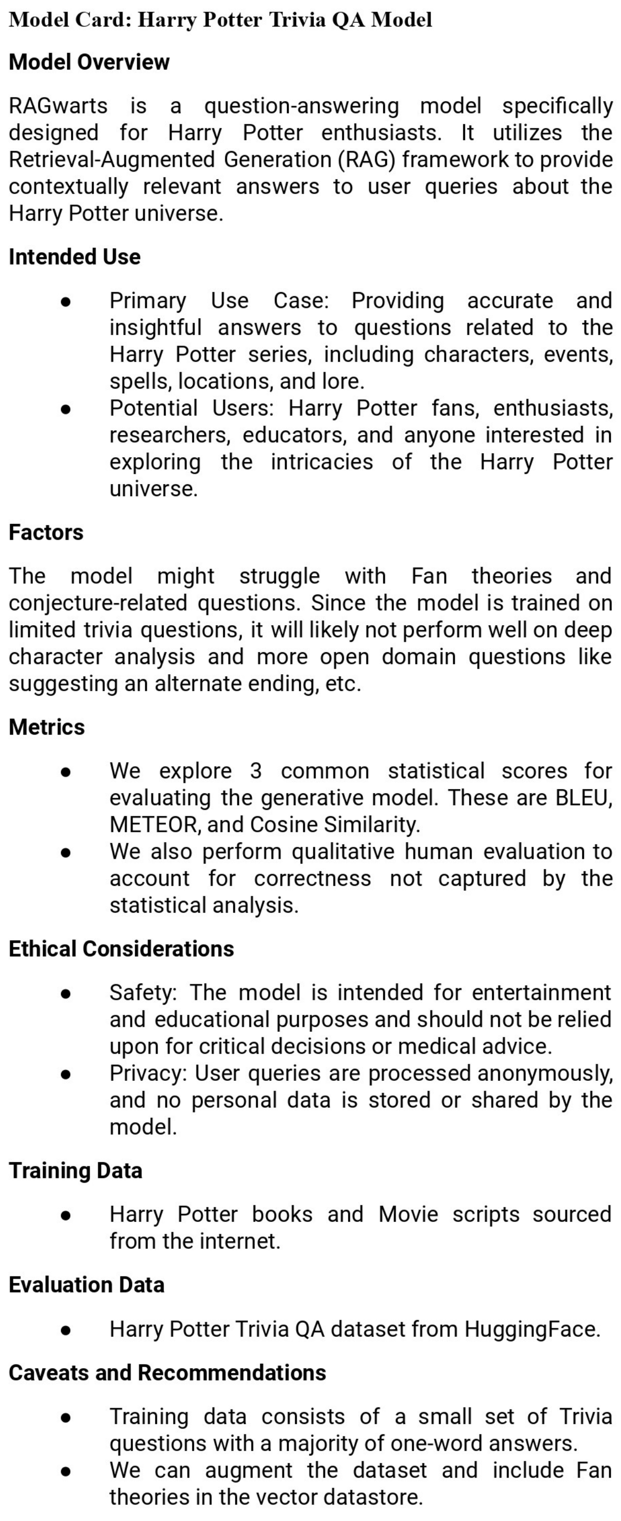
4.4 Fine-tuning

In an attempt to enhance the model's performance, we experimented with continuous fine-tuning, specifically with TinyLlama. We trained it on the Harry Potter books, deviating from the instruction tuning approach. Instead, we fine-tuned it on the book text to predict the next token. However, despite significant time investment, the results were only marginally better than those of the untrained TinyLlama model. Due to the substantial computational resources required and limited improvement observed, we decided against proceeding with Phi-2, recognizing that its larger parameter count would necessitate even more time for training.

5 EVALUATION

For evaluation, we utilized the test split of the Harry Potter Trivia Dataset obtained from HuggingFace, comprising approximately 600 question-answer pairs. During instruction tuning, we recorded the loss and plotted the loss curve to monitor model performance over training epochs. To assess the quality of model-generated answers, we employed four statistical metrics: BLEU score, METEOR score, ROUGE-1 score, ROUGE-L score, and Cosine Similarity Score. Each model was called with the question, and scores were calculated using the model-generated answer and the reference answer. Finally, we computed the average of these scores to gauge the overall performance of the model.

BLEU (Bilingual Evaluation Understudy) score measures the similarity between the generated and reference text based on n-gram overlap. METEOR (Metric for Evaluation of Translation with Explicit Ordering) score evaluates semantic similarity by aligning words and calculating precision and recall. ROUGE-1 score assesses the overlap of unigrams between the generated and reference text, while ROUGE-L score considers the longest common subsequence.



Model Card: Harry Potter Trivia QA Model

Model Overview

RAGwarts is a question-answering model specifically designed for Harry Potter enthusiasts. It utilizes the Retrieval-Augmented Generation (RAG) framework to provide contextually relevant answers to user queries about the Harry Potter universe.

Intended Use

- Primary Use Case: Providing accurate and insightful answers to questions related to the Harry Potter series, including characters, events, spells, locations, and lore.
- Potential Users: Harry Potter fans, enthusiasts, researchers, educators, and anyone interested in exploring the intricacies of the Harry Potter universe.

Factors

The model might struggle with Fan theories and conjecture-related questions. Since the model is trained on limited trivia questions, it will likely not perform well on deep character analysis and more open domain questions like suggesting an alternate ending, etc.

Metrics

- We explore 3 common statistical scores for evaluating the generative model. These are BLEU, METEOR, and Cosine Similarity.
- We also perform qualitative human evaluation to account for correctness not captured by the statistical analysis.

Ethical Considerations

- Safety: The model is intended for entertainment and educational purposes and should not be relied upon for critical decisions or medical advice.
- Privacy: User queries are processed anonymously, and no personal data is stored or shared by the model.

Training Data

- Harry Potter books and Movie scripts sourced from the internet.

Evaluation Data

- Harry Potter Trivia QA dataset from HuggingFace.

Caveats and Recommendations

- Training data consists of a small set of Trivia questions with a majority of one-word answers.
- We can augment the dataset and include Fan theories in the vector datastore.

Figure 1: Model Card

The Cosine Similarity Score quantifies the similarity between vectors representing the model-generated and reference answers based on cosine distance. These metrics collectively provide insights into the quality and relevance of model-generated answers.

We compare the performance of the two baseline models which are TinyLlama and Phi-2 before and after Instruction Tuning on the Question-Answer dataset on the basis of the improvement in the above-mentioned scores. The trend we see after training on five epochs is that the loss is gradually decreasing and the scores have improved by a significant margin given the size of the training set and the number of epochs. The results can be seen in the Radar plot in Figure 2. The untrained model is contained within the trained model metrics. The improvement in BLEU score can be seen in Figure 3. This implies that statistically, the model generated answers with improved similarity with the reference answers. There were some areas where the statistical scores didn't do justice to the model's performance. For example, for the question, "Where could you buy broomsticks?" the model answered "Diagon Alley" whereas the reference answer says "Quality Quidditch Supplies". While on the surface the answer given by the model seems to be incorrect it is not necessarily wrong. Quality Quidditch Supplies happens to be a shop in the Diagon Alley. Statistical scores would rate the answer bad and thus qualitative human evaluation is also necessary. This is just one of the many examples where the model may not match the level of detail of the reference answer.

As observed in the Radar plot, the scores for the phi-2 have almost doubled after the instruction tuning. On the other hand, the scores for TinyLlama have improved marginally as compared to phi-2. This might be because phi-2 is a bigger model than TinyLlama with almost double the number of parameters. Naturally, phi-2's pre-training is superior as compared to TinyLlama and therefore, its untrained performance is also better.

The caveats with the evaluation are mainly the size of the dataset and the type of questions. The dataset has 3k set of question-answer pairs. This size could be increased for better performance. We stuck with this dataset as it was available on HuggingFace and we would need significant computational resources and time to train larger datasets. The second caveat is the type of questions. The trivia dataset contains majority of single-word answer questions or close-ended questions. This means that these models might not do very well on open-ended questions. For example, a character study of Hagrid etc.

5.1 Stastical Evaluation Metrics

5.1.1 ***BLEU (Bilingual Evaluation Understudy Score).*** BLEU is a widely used metric for evaluating the quality of text generated by machine translation systems by measuring the overlap between the generated text (hypothesis) and one or more reference texts. It operates on the principle that the closer the generated text is to a reference text, the higher its quality.

Definition: *BLEU computes the geometric mean of the precision of n-grams of various lengths (up to N) between the hypothesis and the reference texts, while also applying a brevity penalty to penalize*

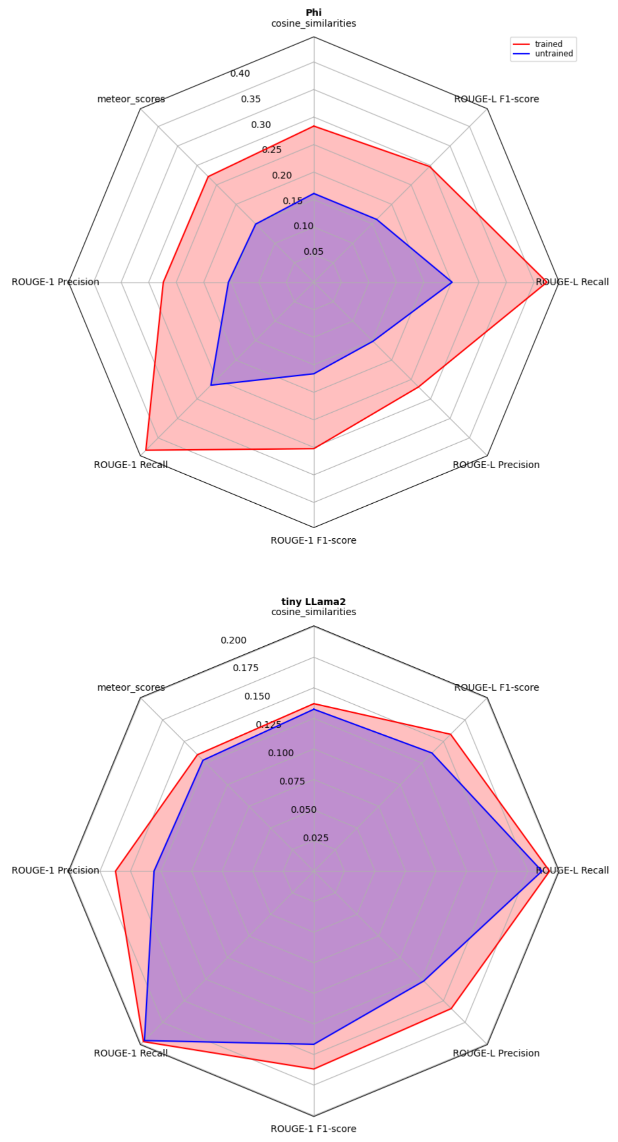


Figure 2: Model performance

overly short generated texts. The BLEU score is expressed as:

$$\text{BLEU} = \exp\left(\sum_{n=1}^{N} w_n \log P_n\right) \cdot BP,$$

where:

- P_n: Precision of n-grams (ratio of matched n-grams to total n-grams in the hypothesis).

- w_n: Weights assigned to n-grams, often uniformly distributed such that $w_n = 1/N$.

- BP: Brevity penalty to penalize outputs shorter than the reference texts.

Components:

(1) ***n*-gram Precision (P_n)**: For a given n, precision is calculated as:

$$P_n = \frac{\text{Number of matched } n\text{-grams}}{\text{Total number of } n\text{-grams in the hypothesis}}.$$

To avoid inflating the score by repeating words or phrases, BLEU uses *clipped counts*, which cap the count of an n-gram in the hypothesis to its maximum count in the reference texts.

(2) **Brevity Penalty (BP)**: The brevity penalty discourages overly short hypotheses that might match reference texts exactly for certain n-grams but fail in length. It is calculated as:

$$BP = \begin{cases} 1, & \text{if } c > r, \\ \exp\left(1 - \frac{r}{c}\right), & \text{if } c \leq r, \end{cases}$$

where c is the length of the hypothesis, and r is the effective reference length (chosen as the closest length to c among the reference texts).

Strengths and Limitations

- **Strengths**:
  - BLEU is simple, computationally efficient, and language-agnostic.
  - It is particularly useful for comparing systems in large-scale evaluations.
- **Limitations**:
  - BLEU does not consider semantic similarity and relies purely on surface n-gram matches.
  - It struggles with capturing nuances in language, such as synonyms or paraphrases.
  - Scores degrade when multiple valid outputs are possible, even if they differ slightly from the reference.

In summary, BLEU provides a fast and widely adopted way to measure translation quality but should be complemented with other metrics for a more holistic evaluation.

5.1.2 *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*.

ROUGE is a set of metrics commonly used for evaluating automatic text summarization and machine translation systems. It measures the quality of generated text by comparing it to one or more reference texts, focusing on recall, precision, and F1-score of overlapping n-grams, word sequences, and word pairs.

ROUGE Variants

The ROUGE family includes several variants tailored to different types of comparisons:

(1) **ROUGE-*n*** (Recall-Oriented n-gram Evaluation):
This variant measures the overlap of n-grams between the hypothesis and reference texts. For n-grams, ROUGE-n is defined as:

$$\text{ROUGE-n} = \frac{\text{Count of Overlapping } n\text{-grams}}{\text{Total } n\text{-grams in Reference}}.$$

- **Recall**: Proportion of n-grams in the reference text that are captured in the hypothesis.
- **Precision**: Proportion of n-grams in the hypothesis that match the reference.

- **F1-score**: A harmonic mean of precision and recall.

(2) **ROUGE-L** (Longest Common Subsequence): ROUGE-L calculates recall and precision based on the longest common subsequence (LCS) between the hypothesis and the reference text. The formula uses:

$$\text{ROUGE-L Recall} = \frac{\text{LCS Length}}{\text{Length of Reference}},$$

$$\text{ROUGE-L Precision} = \frac{\text{LCS Length}}{\text{Length of Hypothesis}},$$

and combines them to compute the F1-score:

$$\text{ROUGE-L F1} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}.$$

Strengths and Limitations

- **Strengths**:
  - ROUGE is intuitive and straightforward, making it widely used for text summarization tasks.
  - It captures recall well, which is particularly important in summarization.
- **Limitations**:
  - ROUGE relies heavily on surface-level overlap and does not account for semantic similarity, such as synonyms or paraphrasing.
  - The quality of the metric can degrade when the reference text allows multiple valid outputs that differ lexically.

In summary, ROUGE is a robust and versatile metric for text evaluation, with its multiple variants offering flexibility for different use cases. However, it is often combined with other metrics to capture nuances beyond lexical overlap.

5.1.3 *Cosine Similarity*.

Cosine similarity is a measure of similarity between two non-zero vectors in an inner product space. It calculates the cosine of the angle between them, which gives a value that indicates how similar the vectors are in terms of their direction.

Given two vectors u and v, the cosine similarity is defined as:

$$\cos(u, v) = \frac{u \cdot v}{\|u\|\|v\|},$$

where:

- $u \cdot v$ is the dot product of the two vectors, which measures the extent to which the vectors point in the same direction.
- $\|u\|$ and $\|v\|$ are the magnitudes (or Euclidean norms) of the vectors u and v, respectively. These represent the lengths of the vectors.
- The result of the cosine similarity lies between -1 and 1:
  - $\cos(u, v) = 1$ indicates that the vectors are perfectly similar and point in the same direction.
  - $\cos(u, v) = 0$ indicates that the vectors are orthogonal (i.e., they have no similarity).
  - $\cos(u, v) = -1$ indicates that the vectors are diametrically opposed, meaning they point in exactly opposite directions.

Strengths and Limitations

- **Strengths**:
  - It is effective in high-dimensional spaces, where the exact magnitude of vectors may not be as important as their direction.
  - It is computationally efficient and does not require the vectors to have the same length.
- **Limitations**:
  - Cosine similarity does not account for the magnitude of the vectors, meaning it only measures the angle between them, which may not always be a sufficient measure of similarity.
  - It is sensitive to the representation of the vectors, and the method of transforming data into vectors can impact the results.

Why Combining Them Provides a Complete Picture:

- **BLEU and ROUGE** focus on different aspects of lexical overlap:
  - BLEU emphasizes precision.
  - ROUGE emphasizes recall.

  Together, they balance the trade-off between the correctness and completeness of the generated text.
- **Cosine Similarity** adds a semantic layer, going beyond exact matches and accounting for the meaning conveyed by the text, which is essential for more advanced NLP tasks like summarization, paraphrasing, or embedding comparisons.
- By using these three metrics in tandem, we get a more holistic understanding of:
  - surface-level accuracy (BLEU),
  - content coverage (ROUGE),
  - semantic similarity (Cosine Similarity),

  making them well-suited for a broad range of text generation and evaluation tasks.
- In summary:
  - BLEU ensures the generated text is accurate.
  - ROUGE guarantees it captures the full scope of reference information.
  - Cosine Similarity measures its overall semantic coherence and similarity.

  Together, they offer a comprehensive, multi-dimensional evaluation.

6 CONCLUSION

In conclusion, this project aimed to develop a question-answering model tailored for Harry Potter fans, leveraging advanced natural language processing techniques such as the Retrieval-Augmented Generation (RAG) framework and instruction tuning. Through experimentation and evaluation, we explored various approaches to fine-tuning models, assessing their performance using statistical metrics such as BLEU, METEOR, ROUGE, and Cosine Similarity scores. While these efforts yielded valuable insights and improvements in model performance, we also encountered challenges and limitations, particularly in balancing computational resources with model effectiveness. Moving forward, this project sets the stage for further refinement and exploration in the intersection of NLP and fandom engagement.

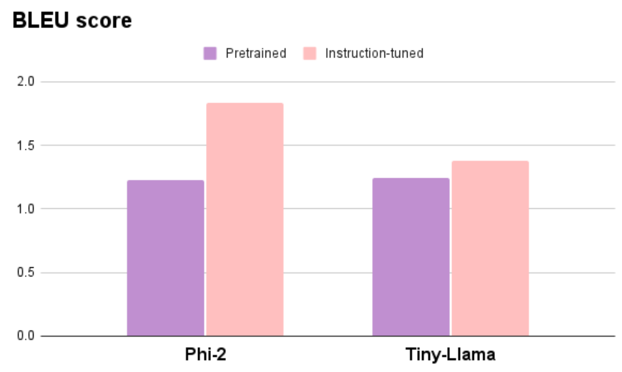


Figure 3: BLEU score

7 CODE

GitHub Repository: https://github.com/MiteshJalan/DiscreteMathProject[10]

REFERENCES

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[3] Mitesh J, Pritish T, and Smit K. Ragwards_dataset_repo. https://github.com/MiteshJalan/Ragwards_Complete_Project, 2024.

[4] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[6] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. pages 228–231, 07 2007.

[7] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019.

[9] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023.

[10] Mahesh Rajput. How to build an intelligent qa chatbot on your data with llm or chatgpt. https://mrmaheshrajput.medium.com/how-to-build-an-intelligent-qa-chatbot-on-your-data-with-llm-or-chatgpt-d0009d256dce, 2023.