# CineSleuth: Detecting Movie Titles from Abstract Plot Descriptions

**FNU Vrinda** vvrinda@umass
**Aman Mehta** amanmehta@umass
**Tejaswini Amaresh** tamaresh@umass
**Nelson Evbarunegbe** nevbarunegbe@umass

## 1 Problem statement

The phenomenon known as ”tip of the tongue,” where individuals struggle to recall movie titles despite vividly remembering plot details, presents a unique cognitive challenge. Our project seeks to harness the capabilities of Large Language Models (LLMs) to bridge the gap between human memory and artificial intelligence. By utilizing LLMs, we aim to transform abstract plot descriptions into accessible cinematic knowledge.

These models excel in pattern recognition within extensive natural language datasets, allowing them to generate responses that mimic the data's style and structure. Our research focuses on leveraging these capabilities by training a language model on a comprehensive dataset of movie plots that spans various genres and eras. This approach addresses the widespread issue of memory retrieval in relation to movies, enhancing how people interact with and recall their cinematic experiences.

We plan to conduct a series of experiments to fine-tune a pre-trained language model using detailed movie plot descriptions and related metadata. The effectiveness of our model will be assessed through rigorous analysis at various stages of training and fine-tuning, aiming to demonstrate significant performance improvements in cinematic knowledge retrieval. This research problem is relatively underexplored in the AI space, and given that our ideation and implementation are not based on existing literature, our project was conducted purely as an experimental endeavor to assess feasibility and establish a proof of concept.

## 2 What you proposed vs. what you accomplished

1. **Enhanced Computational Resources**: We acquired additional computational resources to mitigate the anticipated bottlenecks, enabling us to conduct a broader range of experiments.

2. **Advanced Model Implementation**:

   - Implemented a Sentence Transformer encoder model, improving upon the BERT model initially proposed, to enhance our retrieval-augmented generation (RAG) analysis.
   - Replaced the initially proposed Gemma model with the Mistral model, following insights from recent studies indicating Mistral's superior applicability to our tasks.
   - Employed FAISS, an efficient retrieval system, alongside a baseline cosine similarity retriever, surpassing the basic dense retrieval initially planned.

3. **Dataset Expansion**:

   - Extended beyond the planned Wikipedia movie dataset and crowd-sourced test dataset by incorporating three additional datasets:
     - An iMDB plot dataset, chosen specifically for external validation in our RAG analysis.
     - Two perturbed datasets for experimental analysis which has proved invaluable for testing and is recommended for further exploration in future work.

## 3 Related work

The Tip of the Tongue (ToT) phenomenon, traditionally approached as a retrieval task, has been the subject of extensive study.The approach suggested in (4) utilized the BM25 algorithm, effectively ranking documents based on relevance, mimicking human memory retrieval to help users recall movie titles accurately. To enhance document retrieval, researchers have explored dense retrieval techniques using neural networks to encode queries and documents into dense vectors. (16) showed that these methods outperform traditional term-matching by handling semantic similarities, making them ideal for ToT scenarios. Since efficient data management is crucial in retrieval tasks, Facebook AI Similarity Search (FAISS) (11) proposes a methodology that accelerates similarity searches of dense vectors, improving speed and accuracy in finding relevant movie titles.

The introduction of large language models (LLMs) marked a significant leap in natural language understanding. BERT (Bidirectional Encoder Representations from Transformers) , introduced by (10), revolutionized the processing of textual context by enabling bidirectional analysis. This capability enhanced the comprehension of complex queries and the relationships between words, making BERT particularly effective for natural language processing tasks, including movie title identification. Further advancements were achieved through frameworks such as Mistral (13), which focuses on optimizing and fine-tuning large language models for specific tasks. Mistral's methodologies enhance model performance while reducing computational requirements, leading to more efficient and accurate specialized models. Moreover, (9) proposed a method to adapt LLMs efficiently by reducing their rank, thereby decreasing computational overhead and increasing adaptability. This approach has shown promising results in resource-constrained environments. In situations with limited computational resources, QLoRA (6), focuses on the quantization of LLMs, making them more resource-efficient without compromising accuracy. Additionally, Parameter-Efficient Fine-Tuning (PEFT) (18), adjusts only a small subset of model parameters during the fine-tuning process, enhancing scalability and adaptability to different tasks.

To further improve accuracy, we explored Retrieval-Augmented Generation (RAG) techniques. RAG combines the strengths of retrieval systems and generative models. (8) highlighted that this hybrid approach significantly enhances response accuracy and contextual appropriateness by leveraging both retrieval and generative capabilities. We leave exploration of more advanced developments such as the REAR framework (17) for future work.

Given our task constraints of predicting movie titles for unseen data, we propose evaluating the efficacy of our fine-tuned model by generating synthetic queries—abstract versions of the movie plots the model was trained on. This approach extends the methodology presented in (12), aiming to assess the model's capability in handling abstract prompts and identifying relevant movie titles accurately.

By leveraging these advanced techniques and frameworks, our project aims to significantly improve the performance and accuracy of movie title identification models, ensuring robust handling of a wide range of queries and reliable results for unseen data.

## 4 Your dataset

We have utilized five distinct datasets throughout this project to fine-tune and evaluate our models:

1. **Movie Summary Corpus Dataset** (5): Comprising 42,306 movie plot summaries up to the year 2012 sourced from Wikipedia, and enriched with metadata from Freebase such as character names and their respective actors. This dataset, originating from the Language Technologies Institute and Machine Learning Department at CMU, primarily served as our training data.

2. **IMDb RAG Dataset** (rag): Comprising 45000 movie details it consists of movies released on or before July 2017 that were sourced from IMDb. This dataset served as the external datasource for our RAG search.

3. **Reddit Tip-of-the-Tongue Dataset**: Given the scarcity of datasets tailored to our specific needs, we compiled approximately 176 instances from Reddit communities like r/whatsthemoviecalled (wha) and r/tipofmytongue (tip). Users provided

abstract plot descriptions, and correct movie titles were identified through community responses, qualifying these interactions for inclusion in our evaluation dataset.

4. **Detailed Test Plots**: To assess the model's performance on both seen and unseen data, and to evaluate the impact of detailed versus abstract descriptions, we generated a dataset of perturbed plots of the training dataset. These prompts were subsequently altered to resemble human-described plots using the following prompt with GPT-4.0:

   *"I want to create a tip-of-the-tongue dataset. I will give you detailed Wikipedia plots. If you were a human who couldn't remember this movie and you were asking someone for help to identify the movie based on an abstract description of the plot in a few words, how would you write it? Do not mention the name of the movie anywhere in the plot description. Make sure the query looks humanized and not LLM generated."*

   From these, we randomly selected 199 perturbed plots which had reduced detail levels yet retaining essential elements.

5. **Abstract Test Plots**: Building on the Detailed Test Plots, we further abstracted the same 199 data points into even shorter plot descriptions, around 25-30 words each, using a similar but more restrictive prompt. This dataset aimed to mimic real-life human inquiries and consisted solely of data points previously seen during training.

The challenges associated with these datasets were twofold:

1. The training dataset includes only movies released until 2012, which limits the model's ability to recognize newer releases unless leveraging pre-training knowledge. Continuous updating of the dataset is essential for maintaining relevance.

2. The model's performance is highly dependent on the level of detail in user-provided test plots. While extremely abstract descriptions challenge even human guessing, increased detail significantly enhances prediction accuracy.

**Example:**

1. Movie: "Crazy, Stupid, Love".

   Detailed perturbed plot by GPT: This film is about a middle aged man who learns of his wife's cheating with a coworker. He talks about his divorce in a bar until he attracts a young man who is a womanizer who takes him under his wing to help him rediscover his manhood with hilarious and poignant results. A women named Hannah goes to law school and graduates and expecting her boyfriend to propose to her.

   Highly-abstracted plot by GPT: This film is about a middle-aged man who learns of his wife's cheating on a coworker. He talks about his divorce in a bar until he attracts another man who is a womanizer. Someone in the movie graduates law school.

2. Movie: "The Beast Within". Reddit User plot: I watched a horror movie when I was really young and only remember one scene. Some people were running and trying to hide from the monster. They ended up in a police station and one character locks themselves in the jail cell and backs up against the wall. Then the monster punches through the wall and pulls them through.

### 4.1 Data Pre-processing

People tend to remember movies from more than just the plot, using details like the cast, genre, year of release etc., so we decided to add other metadata available in the training datasets to our input prompt to give the model additional context.

Structure of our modified training prompt:

*Film Genre: <genres>, Starring the actors: <cast names>, Released in the year: <year>, Summary of the movie plot: <Wikipedia plot>*

This modified plot is used as the training context while the movie title is used as the annotation or the true label.

We made similar modifications to all the plots in our IMDb RAG dataset to allow better similarity matching.

## 5 Baselines

For our project, we selected the Mistral pretrained model as the baseline for predicting movie titles from abstract plot descriptions. Mistral, known for its robust language understanding and generation capabilities, is particularly well-suited for this task due to its ability to capture nuanced semantic relationships within text. This makes it a strong candidate for understanding and interpreting abstract plot descriptions and mapping them to the correct movie titles.

To evaluate the performance of the Mistral pretrained model, we conducted experiments on a subset of 200 random data points. These data points were created by perturbing detailed Wikipedia movie plots using ChatGPT-4 to generate more abstract plot descriptions. The evaluation was conducted in two phases:

1. Abstract Plot Descriptions:

   We provided the model with highly abstract plot descriptions.

   Results: The pretrained Mistral model was unable to predict any correct movie titles out of the 200 plots (0/200).

2. Slightly Detailed Plot Descriptions:

   We then provided the model with slightly more detailed plot descriptions, though not as detailed as the original Wikipedia plots.

   Results: The Mistral model correctly predicted 5 out of 200 movie titles (5/200).

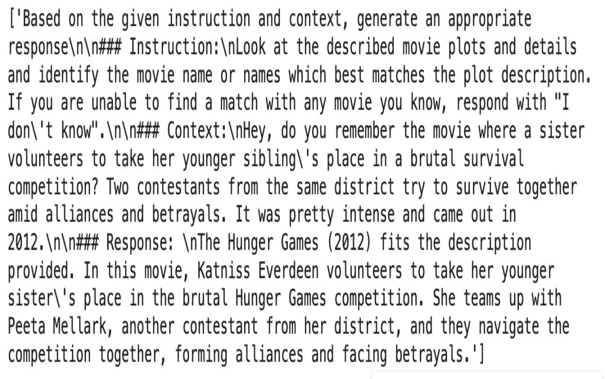

```
['Based on the given instruction and context, generate an appropriate
response\n\n### Instruction:\nLook at the described movie plots and details
and identify the movie name or names which best matches the plot description.
If you are unable to find a match with any movie you know, respond with "I
don\'t know".\n\n### Context:\nHey, do you remember the movie where a sister
volunteers to take her younger sibling\'s place in a brutal survival
competition? Two contestants from the same district try to survive together
amid alliances and betrayals. It was pretty intense and came out in
2012.\n\n### Response: \nThe Hunger Games (2012) fits the description
provided. In this movie, Katniss Everdeen volunteers to take her younger
sister\'s place in the brutal Hunger Games competition. She teams up with
Peeta Mellark, another contestant from her district, and they navigate the
competition together, forming alliances and facing betrayals.']
```

Figure 1: Correct prediction by Pretrained model

3. Test Data collected from Reddit: We then provided the model with the test dataset that we aggregated from Reddit and the pretrained model was able to predict just 3 out of 176 movie titles (3/176).

These results indicate that while the Mistral model has potential, it performs poorly and the performance significantly drops with highly abstract inputs, highlighting the need for fine-tuning and possible enhancements in the retrieval mechanism to improve accuracy.

## 6 Your approach

The goal here is to predict the movie title based on the limited information given to the model. Since most people tend to use plot descriptions or other metadata (like cast, genre, era) to describe a movie, we decided to fine-tune an LLM (Mistral) with a dataset containing the aforementioned data. Once fine-tuning was complete we evaluated the results of the data on our various test datasets to compare performance of the model compared to the baseline.

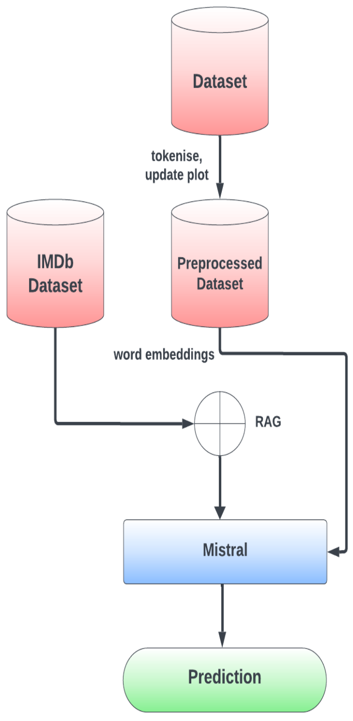


Figure 2: Data Workflow

Since our primary training dataset only included movies from up to 2012, we decided to add an RAG component to the architecture that would not only allow the movie to access newer information at inference time but would also allow the model to see alternate IMDb descriptions of the plot, potentially allowing better prediction. Each of the steps and components have been further detailed below.

### 6.1 Fine-tuning

The first step is to fine-tune our Mistral model on the Wikipedia movie dataset, for which we employed Quantized Low-Rank Adaptation (QLoRA) (6). This approach involves tuning several hyperparameters, namely rank, alpha, learning rate, and batch size. Furthermore, we employ Parameter-Efficient Fine-Tuning (PeFT) to enhance fine-tuning efficiency. PeFT adjusts only a small subset of model parameters, thus making the fine-tuning process more scalable and adaptable to different tasks.

Below, we detail the impact and selection process for each of these hyperparameters:

1. **Rank**: We explored lower ranks in the range [16-32] to fine-tune the model. Lower ranks were chosen to ensure the model's responses were concise and flexible, enabling it to provide single, clear responses without being overly rigid.

2. **Alpha**: The scaling factor that determines the weight merging with the base model. We experimented with alpha values of 16 and 32. The alpha:rank ratio, crucial for determining the scale of weight integration with the base model, was explored using ratios of [32:16, 16:16, 16:32]. We found that an alpha:rank ratio of 32:16 yielded the best results, making the newly learned model weights more prominent and effective.

3. **Number of epochs**: The number of epochs determines the number of times the model runs over the entire training data to revise its weights and gradients. A higher number of epochs is usually preferred to allow the model to learn more complex relationships between the language (without overfitting). We tested our model on 1, 2 and 5 epoch runs depending on the size of the training data being used. Since we had limited compute resources and a large training dataset, we were able to run the model over the entire training dataset for only 2 epochs, but were able to run it for a much smaller dataset for up to 5 epochs.

4. **Learning Rate**: The learning rate, which dictates the step size at each iteration toward minimizing the loss function, was tested with values of 2e-4, 3e-4, and 4e-4. We achieved the best convergence and performance with a learning rate of 3e-4, balancing effective learning with stability.

5. **Batch size**: We evaluated total batch sizes of 8 and 16. The model trained with a batch size of 8 demonstrated superior performance, likely due to more frequent updates and better generalization capabilities with smaller batch sizes.

6. **Gradient Checkpointing**: To optimize memory usage, we implemented gradient checkpointing, which helps in reducing memory requirements by trading off computational overhead.

7. **Max Sequence Length**: We utilized a maximum sequence length of 4096 tokens, accommodating extensive context and improving the model's ability to handle long inputs effectively.

### 6.2 Retrieval-Augmented Generation

RAG is a hybrid model that combines the strengths of language models with the vast knowledge stored in external databases. It enhances language generation tasks by dynamically retrieving relevant document snippets and integrating this information into the generation process, thereby providing contextually richer and more informed outputs. We performed two separate RAG analyses.

#### 6.2.1 Cosine Similarity

The first one being RAG based on cosine similarity selection which involves having the embeddings of the query (i.e. the prompt which requests what movie given a description) compared via cosine similarity to embeddings of the RAG plots, and a FAISS model based on the combination of both dense and sparse RAG. After the models were set up, both RAG approaches were used to select the top k similar plot embeddings to the query, and then appends these to the query to enrich the information for prediction.

The cosine similarity approach first uses an encoder-only model to generate embeddings, in our case we chose the Sentence-BERT model, which uses Siamese BERT networks. According to the model's paper, it is a modification of the pre-trained BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity, which the original BERT model faces massive computational overhead with.

In our context, we have computed cosine similarity of our test query with all movie plots available in the RAG dataset, and then only selected the plots that had similarity greater than a certain threshold to minimise confusing the model with a lot of information. We experimented with how many movies to pick from the RAG dataset and the similarity threshold to understand which works best and it has been detailed in further sections.

The retrieved movies were then used to create a new plot of the template: *<test user plot>, The plot has similarities with the movie(s) <retrieved RAG results>*

#### 6.2.2 FAISS

For the FAISS approach, which is a library for efficient similarity search and clustering of dense vectors. Given a set of vectors, FAISS builds a vector database from these vectors. After the construction of this data structure, when given a new vector (in our case, the query) in dimension, it then compares this to the vector database using the Euclidean distance formula.

Both RAG approaches were implemented solely by ourselves, with library calls from Torch for cosine similarity and top-k similarity identification. We also import FAISS from the FaceBook server for FAISS processing.

We passed the retrieved output of from the retrievers of both Cosine similarity and FAISS and append with the query, which we then pass as the update 'content' section of our pipeline where we call the fine-tuned model to make predictions.

### 6.3 Loss

**Cross-Entropy Loss**: In the context of multi-class classification of the movie titles in this system, cross-entropy loss serves as a fundamental component of the training process. As we aim to predict the correct movie title from abstract plot descriptions, cross-entropy loss enables us to measure the disparity between predicted probabilities and actual labels. By assigning higher penalties for larger deviations from the ground truth, cross-entropy loss guides the model toward more accurate predictions. Through this loss function, the model learns to adjust its parameters to minimize the discrepancy between predicted and true movie titles, enhancing the system's ability to accurately match plot descriptions to the corresponding movies.

$L = -\sum_{j=1}^{C} y_{ij} \log(p_{ij})$

### 6.4 Evaluation

For this task, our goal is to predict the exact movie name. Consequently, the correct answer must be an exact match with the model's prediction. **Accuracy** is the most appropriate evaluation metric for this purpose, as it directly measures the proportion of correct exact matches. Other metrics, such as precision, recall, or F1 score, are less suitable because they account for partial matches, which are not relevant to our requirement of exact matches.

### 6.5 Resources & Libraries

Computers we run our experiments on: We purchased Google Colab Pro three times (after running out of resources severally due to the computational intensity of our experiments) to run our experiments on A100, L4 and T4, utilising 700 compute units. Any use of ChatGPT for prompt perturbation was restricted to ChatGPT 4.0 and the latest ChatGPT 4.0O.

The base libraries used in our study were:

Pandas for pre-processing and dataframe handling, Numpy for some mathematical evaluations, Matplotlib for data visualization, UnslothAI for our transformers and LangChain for FAISS implementation.

There were several implementations of fine-tuning of Mistral and RAG that we used as reference for our code, including Medium articles (V) (Thaker), official documentations (Facebook), as well as research papers. However, they were merely used as reference, and this project is purely an innovation from the very beginning.

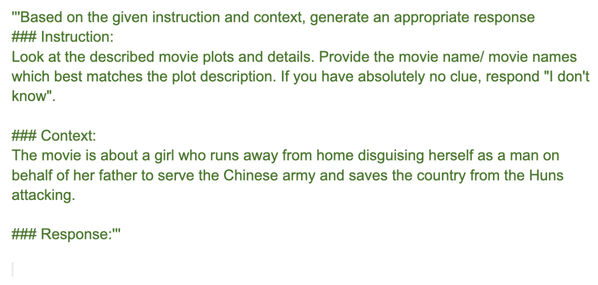
## 7 Experiments:

### 7.1 Prompt Engineering

We experimented with 3 prompts by fine tuning the dataset on a small batch of data containing 200 instances.

Note : We employ the Alpaca style to format the prompts that we provide to the LLMs.

**Example Prompt** :


'''Based on the given instruction and context, generate an appropriate response
### Instruction:
Look at the described movie plots and details. Provide the movie name/ movie names which best matches the plot description. If you have absolutely no clue, respond "I don't know".

### Context:
The movie is about a girl who runs away from home disguising herself as a man on behalf of her father to serve the Chinese army and saves the country from the Huns attacking.

### Response:'''

Figure 3: Example prompt input

The 'Instruction' part of the prompt was experimented for the following texts:

Prompt 1: Identify the movie title through the plot summary described.

Prompt 2: Look at the details of the movie and the plot summary and identify which movie is being described.

Prompt 3: Look at the described movie plots and details and identify the movie name or names which best matches the plot description. If you are unable to find a match with any movie you know, respond with "I don't know".

On testing all the 3 prompts, we finalized Prompt 3 since it resulted in the highest accuracy.

## 7.2 Hyperparameter Tuning

As mentioned in Section 6.1, we experimented with various values of hyperparameters to identify the combination that allowed best model performance:

1. **Alpha and Rank ratio experiments**: Note : The following experiments were conducted with a total batch size of 8 and used 2000 train instances.

    - Alpha = 32, rank = 16, learning rate = 3e-4 , epochs = 5:

      The following accuracy values were observed for each of the test sets:

      Reddit test set : 4/173 = 2.31%

      Detailed Prompts test set : 43/199 = 21.608%

      Concise Prompts test set : 18/199 = 9.04%

    - Alpha = 16, rank = 32, learning rate = 3e-4 , epochs = 5:

      The following accuracy values were observed for each of the test sets:

      Reddit test set : 0/173 = 0%

      Detailed Prompts test set : 13/199 = 6.5%

      Concise Prompts test set : 7/199 = 3.51%

    - Alpha = 16, rank = 16, learning rate = 3e-4 , epochs = 5:

      Reddit test set : 1/173 = 0.57%

      Detailed Prompts test set : 8/199 = 4.02%

      Concise Prompts test set : 3/199 = 1.507%

    Based on the above accuracy results, we decided to finalize on alpha = 32 and rank = 16, making the newly learned model weights more prominent.

2. **Learning rate**:

    Note : The following experiments were conducted with a total batch size of 8 and used 2000 train instances.

    - Alpha = 32, rank = 16, learning rate = 2e-4, epochs = 5

      The following accuracy values were observed for each of the test sets:

      Reddit test set : 1/173 = 0.57%

      Detailed Prompts test set : 8/199 = 4.02%

      Concise Prompts test set : 5/199 = 2.5%

    - Alpha = 32, rank = 16, learning rate = 4e-4, epochs = 5

      The following accuracy values were observed for each of the test sets:

      Reddit test set : 2/173 = 1.15%

      Detailed Prompts test set : 28/199 = 14.07%

      Concise Prompts test set : 11/199 = 5.52%

    The learning rate = 3e-4 resulted in the best model performance.

3. **Batch size**:

    Note : The following experiments were run for 2 epochs and used 30000 train instances.

    - Alpha = 32, rank = 16, learning rate = 3e-4 , batch size = 8

      The following accuracy values were observed for each of the test sets:

      Reddit test set : 6/173 = 3.46%

      Detailed Prompts test set : 18/199 = 9.045%

Concise Prompts test set : 8/199 = 4.02%

- Alpha = 32, rank = 16, learning rate = 3e-4 , batch size = 16
  The following accuracy values were observed for each of the test sets:
  Reddit test set : 3/173 = 1.73%
  Detailed Prompts test set : 13/199 = 6.53%
  Concise Prompts test set : 7/199 = 3.51%

From the above experiments we concluded that alpha=32, rank = 16, learning rate=3e-4 and batch size = 8 would be the optimal hyperparameters for the fine-tuning task.

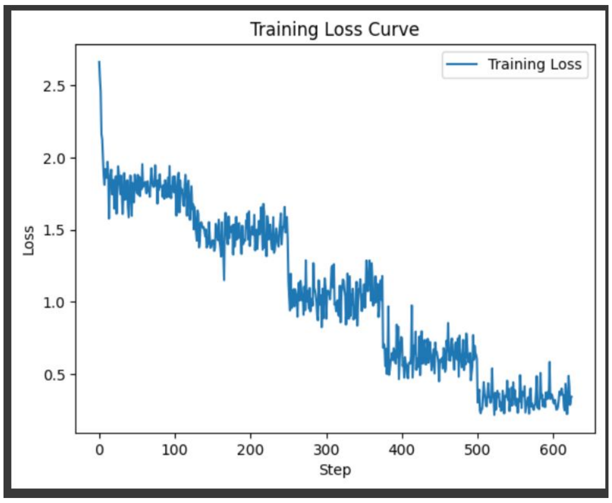


Figure 4: Loss graph for the model trained with learning rate=3e-5, alpha =32, rank=16 and num epochs =5 for 2000 instances

## 7.3 RAG

For the Retrieval-Augmented Generation (RAG), a critical parameter is the number of similar prompts ($k$) selected to augment our user-provided plot descriptions.

### 7.3.1 RAG with Cosine Similarity

To optimize the relevance of retrieved documents, we introduced a similarity threshold in conjunction with cosine similarity. This threshold helps to filter and include only the most pertinent results from the top-$k$ retrieved movies, ensuring the model receives the most relevant contextual information possible.

The following trends were observed from our experiments:

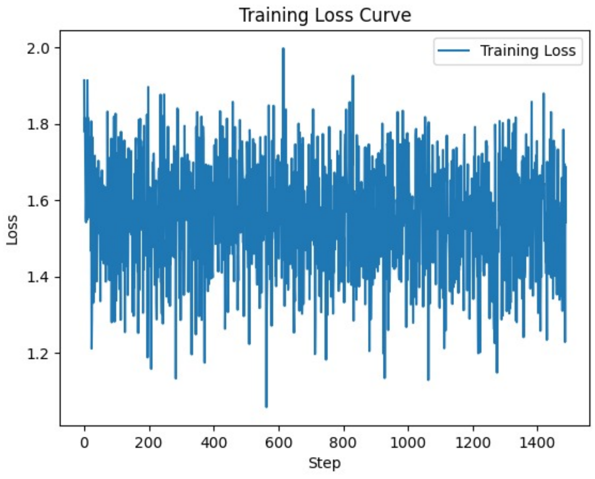
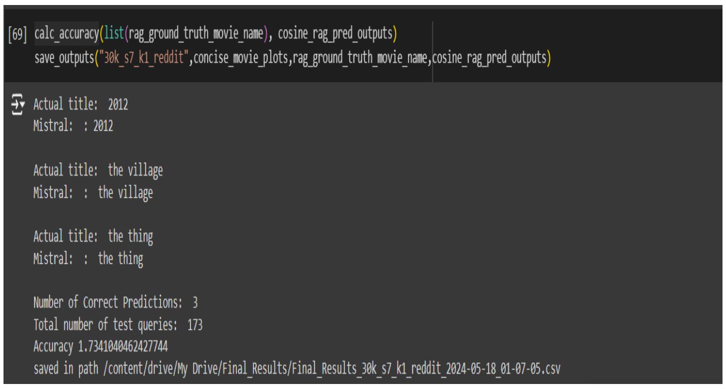


Figure 5: Loss graph for the model trained with learning rate=3e-5, alpha =32, rank=16 and num epochs = 2 for 30000 instances

Table 1: RAG with Cosine Similarity (2000 training instances)

| Dataset | k | Threshold | Correct Predictions | Accuracy |
|---|---|---|---|---|
| Concise | 1 | 0.3 | 1 / 199 | 0.005% |
| Concise | 4 | 0.3 | 4 / 199 | 2.01% |
| Concise | 3 | 0.7 | 6 / 199 | 3.01% |
| Concise | 1 | 0.7 | 6 / 199 | 3.01% |
| Reddit | 3 | 0.4 | 0 / 173 | 0.00% |
| Reddit | 1 | 0.8 | 0 / 173 | 0.00% |
| Reddit | 1 | 0.7 | 1 / 173 | 0.578% |
| Detailed | 3 | 0.7 | 24 / 199 | 12.06% |
| Detailed | 1 | 0.7 | 29 / 199 | 14.58% |



```
[69] calc_accuracy(list(rag_ground_truth_movie_name), cosine_rag_pred_outputs)
     save_outputs("30k_s7_k1_reddit",concise_movie_plots,rag_ground_truth_movie_name,cosine_rag_pred_outputs)

Actual title:  2012
Mistral:  : 2012

Actual title:  the village
Mistral:  :  the village

Actual title:  the thing
Mistral:  :  the thing

Number of Correct Predictions:  3
Total number of test queries:  173
Accuracy 1.7341040462427744
saved in path /content/drive/My Drive/Final_Results/Final_Results_30k_s7_k1_reddit_2024-05-18_01-07-05.csv
```

Figure 6: Improved performance with RAG

1. RAG integration showed minimal impact on models trained with 2,000 instances, occasionally reducing performance compared to using only the fine-tuned model.

2. For models trained on 30,000 instances, RAG

Table 2: RAG with Cosine Similarity (30,000 training instances)

| Dataset | k | S.T | Predictions | Acc. |
|---|---|---|---|---|
| Concise | 4 | 0.3 | 2 / 199 | 1.01% |
| Concise | 3 | 0.7 | 7 / 199 | 3.52% |
| Concise | 1 | 0.7 | 8 / 199 | 4.02% |
| Concise | 1 | 0.3 | 9 / 199 | 4.52% |
| Reddit | 3 | 0.4 | 0 / 173 | 0.00% |
| Reddit | 1 | 0.8 | 3 / 173 | 1.73% |
| Reddit | 1 | 0.7 | 3 / 173 | 1.73% |
| Detailed | 3 | 0.7 | 32 / 199 | 16.08% |
| Detailed | 1 | 0.7 | 33 / 199 | 16.58% |

integration generally enhanced or maintained performance levels, particularly improving results for models that were less effectively trained.

3. A balance in the value of $k$ and the similarity threshold proved crucial; overly inclusive or exclusive settings tended to degrade performance. Optimal settings typically involved a higher confidence in the selected $k$ value, often at $k = 1$.

The use of RAG based on Cosine similarity appeared less effective in aiding well-fine-tuned models, possibly due to the introduction of noise or irrelevant information. However, it significantly improved performance in models that were weakly trained. This supports the practice of using RAG with pre-trained models, as it supplements these models with additional relevant information without the risk of overwhelming them.

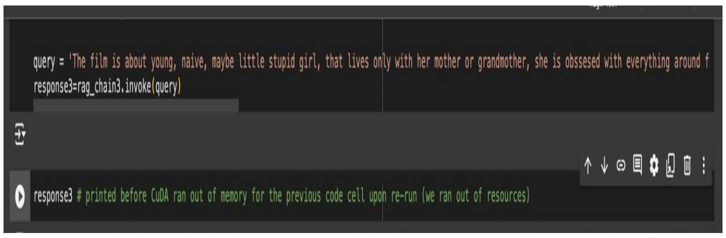
**Advantage:** Cosine similarity is computationally efficient, making it scalable for large datasets.

**Drawback:** It only considers angular distance between vectors, neglecting magnitude, which can compromise retrieval quality when dealing with diverse document lengths and semantic richness, as it overlooks term frequency variations and synonymic relationships.

### 7.3.2 RAG with FAISS

Facebook AI Similarity Search(FAISS) works by efficiently searching for nearest neighbors in large datasets, which can be crucial for tasks like retrieval-augmented generation (RAG). Unlike cosine similarity that calculates an angular score between vectors, FAISS employs specialized indexing and quantization techniques to handle large-scale vector databases. This allows FAISS to perform high-speed similarity searches, quickly retrieving the most relevant document vectors from a vast pool. When integrated into RAG, FAISS enables the model to access the most contextually pertinent information from the dataset in a fraction of the time it would take using traditional search methods, enhancing both the efficiency and effectiveness of the generation process. However, FAISS is extremely compute heavy and it was not feasible to run the FAISS experiments on Colab without running into a GPU CUDA Out of Memory error while trying to create the index over the entire RAG dataset.

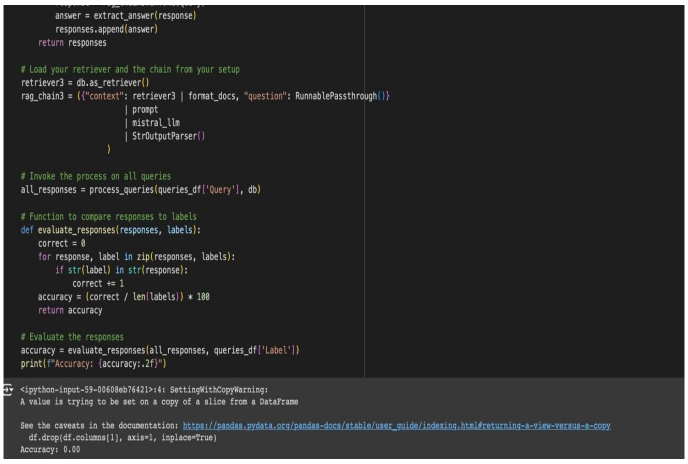
For the sake of experimentation, we built an FAISS index over 300 points in the RAG dataset (anything over that was giving a GPU error) and used an RAGChain setup to retrieve the most similar movies and then prompted the LLM with the updated plot.



```
query = 'The film is about young, naive, maybe little stupid girl, that lives only with her mother or grandmother, she is obssesed with everything around f
response3=rag_chain3.invoke(query)

response3 # printed before CuDA ran out of memory for the previous code cell upon re-run (we ran out of resources)
```

Figure 7: RAG with FAISS



```
        answer = extract_answer(response)
        responses.append(answer)
    return responses

# Load your retriever and the chain from your setup
retriever3 = db.as_retriever()
rag_chain3 = ({"context": retriever3 | format_docs, "question": RunnablePassthrough()}
                     | prompt
                     | mistral_llm
                     | StrOutputParser()
                  )

# Invoke the process on all queries
all_responses = process_queries(queries_df['Query'], db)

# Function to compare responses to labels
def evaluate_responses(responses, labels):
    correct = 0
    for response, label in zip(responses, labels):
        if str(label) in str(response):
            correct += 1
    accuracy = (correct / len(labels)) * 100
    return accuracy

# Evaluate the responses
accuracy = evaluate_responses(all_responses, queries_df['Label'])
print(f"Accuracy: {accuracy:.2f}")
```

```
<ipython-input-59-00608eb76421>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(df.columns[1], axis=1, inplace=True)
Accuracy: 0.00
```

Figure 8: RAG Accuracy with FAISS

As expected, we do not see good results with FAISS as we restricted the dataset to a very small number forcing the model to pick results that may be incorrect. However, we found FAISS to be significantly quicker while making inference and can presume to get stronger results with better compute resources.

## 8 Error Analysis

The following cases involved the model failing to predict correct movie titles. The annotated file can be found in the folder named ”error_analysis”.

### 8.1 Analysis:

- Instance 1:

  Prompt: The female lead froze sperm of someone she got from someone else to get back at a man then used it on herself and says its his he looks surprised and she says ”Ask me how I did it”

  Actual Title : Gone Girl

  Predicted Title : She’s on vacation

  Analysis : In Fig.6, due to the poor prompt description, we observe that the model hallucinates and predicts a title that is not accurate but closely related to something about a female. Since the plot contains only details of a very small scene in the movie, it is hard to predict accurately from a model that has been fine tuned on movie plots that do not contain each and every scene of the movie.

- Instance 2:

  Prompt: whats the movie where there’s a lot of people on a cruise ship(?) and they’re all dancing and they all get sliced in half by a freak accident involving a wire?

  Actual Title : Ghost ship

  Predicted Title : Titanic

  Analysis: In this instance, the model incorrectly predicts the movie title. This error occurs due to the model’s difficulty in distinguishing between plots with similar settings. Both movies involve significant events taking place on large ships, but the nature of the accidents differs greatly. ”Ghost Ship” features a gruesome scene where a wire slices through a group of people, whereas ”Titanic” is known for the ship hitting an iceberg and sinking. The model’s confusion highlights the challenge of accurately classifying movies with overlapping themes or settings.

- Instance 3:

  Prompt : Can you help me identify a movie from its plot? It’s a 1973 romance drama that explores themes of moral weakness and temptation. The story starts with a Sanskrit saying about desire and follows the life of Jayanthi and her soldier-husband, who returns from war physically and emotionally scarred. The narrative intensifies with the introduction of their brash, young neighbor Nanjunda, leading to a complex emotional entanglement marked by infidelity and internal conflict. The setting, suggestive of Kodagu with its undulating hills, adds a picturesque backdrop to this tale of passion, betrayal, and moral dilemma.

  Actual Title : Edakallu Guddada Mele

  Predicted Title : <Actual Title in Kannada Script>

  Analysis : Here, although the model correctly predicts the movie title, evaluating multilingual content is a challenge, due to which the above prompt is marked as an incorrectly classified instance.

Based on our observations concerning the performance of Retrieval-Augmented Generation (RAG) using Cosine Similarity, it is evident that the characteristics of the dataset significantly influence the matching quality. Specifically, the IMDb dataset features plot descriptions that are phrased differently and are generally more concise compared to those in Wikipedia. This discrepancy in description style and length has notable implications for the effectiveness of our model. We observed improved matches when the Wikipedia plot descriptions were comparably shorter, the input plots were sufficiently detailed, or the plots contained highly distinctive elements. For example, the movie ”2012” is characterized by specific and unique plot points, which makes it more easily identifiable by our system. This suggests that the distinctiveness of content and the level of detail are crucial factors in achieving accurate and reliable retrieval results with RAG, especially when handling datasets with varying narrative styles and summary lengths.

### 8.2 Inference on Analysis:

Based on the observed results and trends, we analysed and brainstormed the potential reasons behind the poor performance of our models despite several rounds of hyperparameter tuning alongside several other experiments. The reasons have been detailed below:

1. Unseen movies/newer movies that were not present in the training set and RAG dataset will never be predicted correctly. This limitation can be alleviated if the RAG data source could be an online real time up-source where movies are being updated on-the-fly. This would be a good avenue for future works.

2. We had limited training and compute resources, which is our central bottleneck. We were able to observe a trend of improved prediction accuracy's as we increased the training rounds.

3. Another bottleneck we faced was the size of the dataset. Training such large language models demand large datasets to be effective. Due to resources available, we limited to the a few thousands which is yet reasonable but still not as effective as having more.

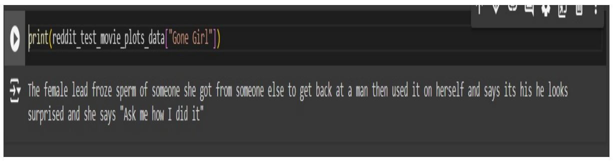
4. More work could be done in the prompt tuning part of the GPT. This could potentially lead to improvements in the evaluation results.

5. We faced the bottleneck of having very similar movie plots and the model then predicting one movie for the other. For example, the Titanic movie plot is very similar to the Iceberg movie plot and our model was guessing Iceberg for Titanic. We suspect this is due to the level of detailing in the user prompt and the more information that is provided, the lower are the chances of incorrect prediction.

6. Humans could use other information such as scene and object descriptions to identify a movie (as observed in the Reddit Test dataset). However, LLMs are not intelligent and can only restrict their understanding to the Wikipedia plot that they were trained on. This is an innate drawback of LLMs which is strongly visible here as it does does not take into consideration human emotions and specific descriptions that could help prediction.

7. We only have a binary evaluation metric of accuracy here since the model is predicting only one movie title, we can only check for an exact match between the predicted movie and the actual movie. If the model could generate multiple possible suggestions (like a recommender system) then there is a possibility of finding higher matches. This would require a more complex architecture where we would have to extract internal embeddings of the model to identify other potential candidates.

8. Amongst humans, there can be multiple iterations of negative and positive feedback for a particular input to identify a movie, however, our setup is not built to take any negative feedback and is only designed to make predictions. As mentioned earlier, in the 'Gone Girl' example in Fig.6, a poor prompt causes the model to make an incorrect prediction.



```
print(reddit_test_movie_plots_data["Gone Girl"])
```

```
The female lead froze sperm of someone she got from someone else to get back at a man then used it on herself and says its his he looks
surprised and she says "Ask me how I did it"
```

Figure 9: Poor Quality Input

## 9 Contributions of group members

Each member was involved to a reasonable amount on each of the tasks performed, as was anticipated from the proposal. We however had each member be in full charge of a specific section and others assist in areas of debugging, troubleshooting and brainstorming to ensure everyone is well-informed on each subsection of the project. Below are the leads of each section:

- FNU Vrinda: Project lead and coordinator, experimental data collection, data pre-processing, hyperparameter tuning, performing experiments, report writing.

- Aman Mehta: Project conceptualization, model development and chaining, model evaluation lead, data pre-processing, report writing.

- Tejaswini Amaresh: Fine-tuning subproject lead and developer, test data collection, experiments, hyperparameter tuning, report writing.

- Nelson Evbarunegbe: RAG subproject lead and developer, data curation from GPT4, test data collection, report writing.

## 10 Results

The best accuracy we observed was for the model that was trained for 5 epochs, for 2000 instances with the hyperparameters of rank = 16, alpha = 32, learning rate = 3e-4. The accuracies observed for each of the test sets were as follows:

Reddit test set : 4/173 = 2.31%

Detailed Prompts test set : 43/199 = 21.608%

Concise Prompts test set : 18/199 = 9.04%

Although the model that was trained on 2 epochs for 30000 instances could predict 6/173 test plots from the reddit dataset, it performed poorly when detailed and concise versions of the train plots were provided as test inputs. Therefore, the smaller model that was trained for more epochs can be interpreted as the best model.

If we train the model with 30000 instances for higher number of epochs, we're very likely to obtain results with high accuracy.

## 11 Conclusion

Our project employs a diverse range of advanced techniques to tackle the problem of predicting movie titles from abstract plot descriptions. We utilized various transformer models, including encoder-only models (Sentence-BERT), encoder-decoder models (Mistral), and parameter-efficient fine-tuning methods (LoRA). Additionally, we explored cutting-edge methods such as retrieval-augmented generation (RAG) using cosine similarity and FAISS-based retrieval, extending our approach beyond conventional methodologies.

Although our initial setup did not yield exceptional results, our various experiments indicate a clear trend: with more detailed inputs and enhanced computational resources for extended training, coupled with advanced architectural improvements, we can achieve better performance. The nature of the problem, which demands high factual accuracy and struggles with newer movie predictions, suggests that modifying our problem statement to predict the closest matches instead of exact movie titles could be more effective. This approach would transform our model into a recommender system, providing valuable suggestions even when an exact match is not found.

## 12 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used. Yes, we primarily used ChatGPT (GPT 3.5 and 4).

*If you answered yes to the above question, please complete the following as well:*

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - **Data augmentation**

    Prompt: Create variations of a movie plot to simulate different levels of abstraction.

    Prompt: I will give you 200 detailed Wikipedia plots and I want to create a tip-of-the-tongue dataset. if you were a human who couldn't remember this movie and you were asking someone for help to identify the movie based on an abstract description of the plot in 20-50 words at max, how would you write it? Do not mention the name of the movie anywhere in the plot description. Make sure the query looks humanized and not LLM generated.
  - **Prediction**

    Prompt: Identify the movie title through the plot summary described.

    Prompt: Look at the details of the movie and the plot summary and identify which movie is being described.

    Prompt: Look at the described movie plots and details and identify the movie name or names which best matches the plot description. If you are unable to find a match with any movie you know, respond with "I don't know".

    Prompt: Provide a detailed explanation of classification and retrieval-based metrics suitable for evaluating the movie guesser system.

- **Model and Fine-Tuning**

  Prompt: Explain why the Mistral model is particularly suitable for predicting movie titles from abstract plot descriptions

  Prompt: What is the default loss function used by the unSloth library for fine-tuning models?

  Prompt: How can we modify or replace the default loss function in the unSloth library?

  Prompt: Explain how to set up checkpoints and validation steps in unSloth.

  Prompt: How can we reduce overfitting during the fine-tuning process using unSloth

- **RAG**

  Prompt: Explain RAG and its benefits for enhancing context retrieval

  Prompt: Describe how RAG can be integrated into the movie guesser system to improve predictions.

  Prompt: Explain how to use FAISS to build an index of movie plot embeddings for fast retrieval

  Prompt: Describe the use of cosine similarity in ranking candidate movie titles based on their relevance to the input plot description

- **Report**

  Prompt Give me 1-2 lines to describe RAG

  I need to write the problem section statement where I need to Summarize the goal of your project and its motivations in this section. This is my content: <content>Can you help rewrite it formally?

  Other prompts to rewrite certain sections of the report more formally.

## References

[rag] The movies dataset.

[tip] Tip-of-my-tongue.

[wha] What's that movie called.

[4] Arguello, J., Ferguson, A., Fine, E., Mitra, B., Zamani, H., and Diaz, F. (2021). Tip of the tongue known-item retrieval: A case study in movie identification. *arXiv preprint arXiv:2101.07124*.

[5] David Bamman, B. O. and Smith, N. A. (2013). Learning latent personas of film characters.

[6] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

[Facebook] Facebook. Faiss tutorial.

[8] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2022). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[9] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[10] Jacob Devlin, Ming-Wei Chang, K. L. K. T. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

[11] Jiang, Y., Lin, D., Li, R., and Zhu, X. (2017). Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

[12] Mysore, S., McCallum, A., and Zamani, H. (2023). Large language model augmented narrative driven recommendations. *arXiv preprint arXiv:2306.02250*.

[13] Tang, Z., Jiang, Y., He, Z., Chen, H., Tan, L., Liu, F., and Lin, Z. (2021). Mistral: Optimizing retrieval and fine-tuning large language models for open-domain question answering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1777–1786. ACM.

[Thaker] Thaker, M. Ragchain.

[V] V, K. Fine-tuning large language models with unsloth.

[16] Vladimir Karpukhin, Barlas Oğuz, S. M. P. L. L. W. S. E. D. C. W.-t. Y. (2020). Dense passage retrieval for open-domain question answering.

[17] Wang, Y., Ren, R., Li, J., Zhao, W. X., Liu, J., and Wen, J.-R. (2024). Rear: A relevance-aware retrieval-augmented framework for open-domain question answering. *arXiv preprint arXiv:2402.17497*.

[18] Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., and Wang, F. L. (2024). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.