Augmented Semantic Search using Text Summarization

Chaitanya Deshpande
University of Southern California
cdeshpan@usc.edu

Divya Srikant
University of Southern California
dsrikant@usc.edu

Namita Mutha
University of Southern California
nmutha@usc.edu

Sanket Sapkal
University of Southern California
sapkal@usc.edu

Vasudha Mahajan
University of Southern California
vasudhab@usc.edu

Abstract

With an extensive collection of movies available online, there arises a need for a context-based movie search engine. Currently, the movie search on Prime, Netflix, or HBO Max is limited to movie titles, cast, or genres. However, the user would have a better search experience if they could search for movies using a plot based query (For example, a search for "romantic movie set in Europe where the girl has cancer" would return the result: "The fault in our stars"). NLP techniques like semantic search and text summarization can be applied to address this problem statement. The project aims to overcome this limitation and enhance the search experience by implementing a context-based semantic search engine.

1 Introduction

Semantic search is a data searching technique in which a search query aims to find keywords and determine the intent and contextual meaning of the words a person uses for a search. The semantic search comes into play when users don't know what exactly they are looking for. This applies to movie searches as well. Often, the human brain is wired to remember the crux of a movie more than its actual title. People prefer to search with whatever information they remember rather than the precise details (like a movie name or genre).

Semantic search is better equipped to handle such queries in natural language rather than relying on keyword matching. Instead of picking up keywords to parse search results, semantic search, when performed on plot data, will interpret the keyword in a contextual manner. For instance, if one were to search for a "dreamy family vacation," it would equate dreamy to ideal (not imaginary) as it better fits the context. It is difficult to convert such queries in a manner that a computer might understand.

Movie plots are summarized by human authors, and these brief plots are bound to miss fine details of the movie. The authors can even have different perspectives towards the movie, resulting in different plot descriptions. It often happens that some part of the movie might be appealing to the audience which is not relevant to the plot. A brief but exegetic description of the movie can be generated by summarizing the movie's dialogues. Search results for a movie query can be further improved by performing a semantic search on the summarized movie dialogues instead of the movie plots.

The main objective of this project is to perform a comparative study on results returned by a semantic search on the movie plot vs on the dataset generated by summarizing the movie dialogues/subtitles.

2 Related Work

Earlier work on semantic search for movie search involved the usage of word embeddings, POS tagging, and NER [1]. Synthetic queries can be generated using semantic query labeling in the movie domain. These synthetic queries are an excellent alternative to context extraction from a query. Search results can be improved with synthetic queries only when the synthetic query is relevant to the original query given by the user. However, there has been a relatively low exploration of the usage of large language models in semantic search [2]. Usage of LexRank and LSA algorithms has been explored in the previous works for summarization of films and documentaries [3].

We did not come across any movie search engine using a query for movie plots during the literature survey, and we are trying to create one such engine. We plan to compare the results of semantic search on movie plot data vs. semantic

search on summarized movie dialogues to check if we are achieving the same or better results (results obtained by comparing the IMDB rating of the movie). This project improves upon the above-mentioned related works by exploring the usage of large language models for generating document embeddings for semantic search and document summarization. This project also improves upon the similarity search performance by using FAISS.

3 Dataset

3.1 Data Preparation

The dataset consists of a publicly available movie plot which contains details of several movies from around the world scraped from Wikipedia articles (`https://www.kaggle.com/code/sbrvrm/semantic-search-with-sbert-faiss/data`) provided by Kaggle. The dataset consists of 34,886 rows with the columns - *Release Year, Title, Origin/Ethnicity, Director, Cast, Genre, and Plot*. The release years of movies range from 1901 to 2017. The project's scope is English-origin movies and after filtering on the basis of language we have a dataset of 22,346 rows. Three more columns, namely - *IMDbID, IMDbRating* and *SummarizedSubtitles*, were also added to the dataset using the following methods.

IMDb ID and Rating

- Movie's IMDb ID is needed for fetching its subtitles. Additionally, IMDb rating is needed to perform a comparative analysis of the results. A python script was written which uses OMDb API which queries IMDb using movie title and fetches the imdbID, imdbRating, imdbVotes, Year, Awards etc.
- For each movie in the dataset, we populate the IMDbID and IMDbRating columns using this data. If a movie is not present in the IMDb database, we remove it from the original dataset.

Subtitles

Movie subtitles are fetched primarily from Opensubtitles.org, and if a particular movie's subtitles do not exist, the movie is ignored to maintain consistency of the subtitles. A python script fetches the movie subtitles using its IMDb ID from the above mentioned website.

Data Filtering

Out of the 22,346 movies, the movies were filtered based on several criterias as follows:

- Only American movies were selected thus reducing the dataset size to 17,377.
- As movies released before the year 2000 did not have good quality subtitles, only the movies released after the year 2000 were considered further reducing the dataset size to 3785..
- Fetched IMDb rating for all movies and sorted them in order of highest to lowest.
- Summarizing each movie subtitle took 30-40 minutes on Google Colab using a GPU hardware accelerator. Due to limited resources and time availability to find the right model for unsupervised text summarization that suits the need of the project, we sampled the dataset of each IMDB rating bucket (e.g. 4.0 - 5.0, 7.0 - 8.0 and 8.0 - 9.0) to get a randomized dataset and selected 150 movies.
- Fetched subtitles for 150 movies and ignored those which did not have subtitles. Some movies did not have good quality subtitles and those were manually checked and removed. The final dataset obtained had 80 movies.

3.2 Data Cleaning

The following data cleaning procedures were performed on the summarized subtitles:

- Removed extra spaces.
- Removed any HTML tags present in the subtitles.
- Removed URL that might occur in the textual data.
- Removed emojis/symbols and special characters like $, #, %, (,), {, } etc.
- Removed non related sentences from subtitles like "downloaded from Opensubtitles.org".

4 Design

The system takes a text query as an input and gives an output list of top 3 movie titles and IMDb ratings which are most relevant to the user description. For example, a search for "monkeys are stuck in space and escape to earth" would output movie titles like "War for the Planet of the Apes", "WALL-E", "Space Chimps".

4.1 Subtitle Summarization

The subtitles of the movie are summarized using a pre-trained transformer model - "*facebook/bart-large-cnn*". The model performs supervised and extractive text summarization on the movie subtitles. Extractive summarization technique was used because it produced better summaries than abstractive summarization. Subtitles are summarized in the following steps (as shown in Figure 1):

- The subtitle text is divided into individual sentences using a sentence tokenizer. Later, the sentences were combined to form a chunk which has no more than 1022 words. A list with multiple such chunks is created.
- Each chunk is then summarized using "*facebook/bart-large-cnn*", all chunk summaries are combined together to create the complete subtitle summary for a movie. The summary of a chunk can be between 30 to 100 words.
- Steps 1 and 2 as described above are performed for each movie and the summary was added to the dataset.

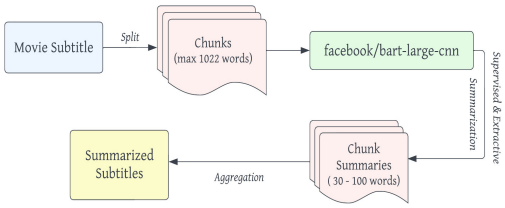

Figure 1: Subtitle Summarization Pipeline

4.2 Movie Plot and Summarized Subtitle Embedding

Searching for a movie based on the movie plot is an asymmetric search. The search is performed over long text, long plots or long dialogue summaries in our case. Thus, a dot product model like "*msmarco-distilbert-base-v3*" [4] is better suited for long text retrieval. We use "*msmarco-distilbert-base-dot-prod-v3*" sentence transformer model which converts each movie plot and summarized dialogues to 768 dimernsional vector embeddings. The max length of sequence of the transformer is set to 512. We generate 2 models: one from encoding movie plots and the other from encoding movie subtitle summary.

4.3 Search Query Embedding

The search query can be provided in the notebook. The search query embedding is generated using the same model as step 3: using "*msmarco-distilbert-base-v3*". The generated embedding is a 768 dimensional vector.

4.4 Search Result Retrieval

The generated movie plot embedding and movie summary embedding are indexed using the library: FAISS (Facebook AI Similarity Search). The search query embedding is used to perform a hash based search (using FAISS) on movie plot embeddings and movie subtitle summary embeddings and top 3 movie titles are returned from each of these. The semantic search architecture can be seen in Figure 2.

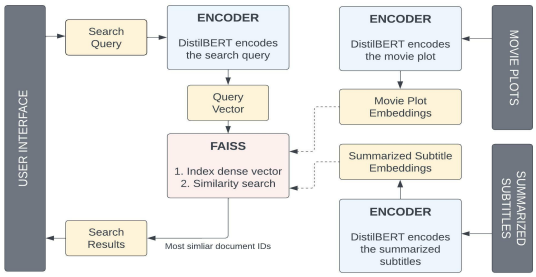

Figure 2: Semantic Search Architecture

4.5 Fine Tuning

The sentence transformer model could not be tuned directly due to the absence of existing supervised dataset mapping between queries and movie plots/dialogues. Thus, an unsupervised learning approach was used to generate synthetic queries and these queries were used to fine tune the model. Synthetic queries are generated separately for both movie plot and movie subtitles.

4.5.1 Synthetic Query Generation

For each movie plot and summarized subtitles possible queries were generated in the form of questions.

- The number of queries to generate from every paragraph was set to 5, max length for each paragraph was set to 512, max length of query to be generated was set to 64.
- Chunk for each movie plot/subtitles was used to generate 5 synthetic queries.
- These pairs of generated queries and paragraphs were used to fine tune the sentence transformer model.

- The fine tuned model was able to capture the semantic and syntactic information of these pairs.

4.5.2 Bi-Encoder

- The same model "*msmarco-distilbert-base-dot-prod-v3*" used in section 3 was used for generating embeddings.
- These embeddings are used for adding the mean pooling layer which was passed to the cosine similarity layer for retrieval.
- MultipleNegativesRankingLoss loss function was used as only positive pairs were considered and it helped in better sentence representations.
- Using these synthetic queries and bi-encoder the model was fine tuned.

Movie subtitles are summarized as described in Step 1 above and added to the dataset. Then we generate plot embeddings and subtitle summary embedding as described in Step 2. Later the search query is embedded as described in Step 3 and the search results are retrieved using FAISS (Step 4).

5 Experimental Setup

5.1 Dataset

Open source dataset consisting of several movies (Wikipedia Movie Plot data) was used. For each movie, subtitles are scraped from Opensubtitles.org and IMDb rating was scraped as well. The dataset was augmented with IMDb Rating, IMDB Id, Subtitles and Summarized subtitles for the movie.

5.2 Model

A pre-trained "*facebook/bart-large-cnn*" model was used for extractive summarization of subtitles and "*msmarco-distilbert-base-dot-prod-v3*" sentence transformer model was used for generating document embeddings of movie plots and summarized subtitles. Further, the fine-tuned model was able to capture the semantic and syntactic information of the data.

Google Colab (with GPU hardware accelerator) was used to host and execute the notebook. The dataset was hosted over google drive for easy access in the notebook. The synthetically generated queries and the fine-tuned model are also stored on Google Drive.

5.3 Version Control

The repository was hosted on Github to track changes made in the notebook, scraping and data cleaning scripts.

5.4 Evaluation

Test data was generated by manually reading the movie plots and forming queries which consist of top 3 movie titles (as per imdb rating) as output for each test query. These queries were used as input for testing the model accuracies. The results from both the models were compared against expected movie titles. Hit and miss strategy was used to calculate the accuracy of the model.

6 Results And Discussion

Accuracy turned out better with summarized subtitles embeddings rather than movie plot embeddings. This is because, summarized subtitles are feature rich (i.e. contain more information about the movie) because it captures the emotion, surroundings and tone of the scene which is more informative than just the plot. Fine-tuning both the models (plot and subtitles) by passing the synthetically generated queries to the Bi-Encoder model resulted in improved accuracy.

Models	Plot embeddings	Summarized Subtitles Embeddings
Non-Fine-tuned model	0.363	0.454
Fine-tuned model	**0.575**	**0.727**

Table 1: Results

Summarized subtitle embeddings also resulted in movies with higher IMDB ratings than movie plot embeddings when the query had more than 3 possible results. Project Link: `https://github.com/SanketSapkal/semantic_search`

6.1 Applications Of The Project

- Find recipes based on ingredient or spice descriptions.
- Find novels from queries. E.g. User can input query: "Fantasy magic novel" for "Harry Potter"
- Legal language recommendation for legal document generation. E.g. user can search for "Standard NDA clause with early termination" and get the required legal language.

7 Future Work

To enhance the results and the search experience further, use a bigger dataset with at least 100

movies from each genre so that there are enough results for each user query.

Explore movie search using popular dialogues in the movie.

8 Division Of Labour

One person was responsible for scraping and fetching the movie subtitles and IMDb ratings. One person was responsible for data cleaning. One person was responsible for performing extractive text summarization on the movie subtitles. One person was responsible for building the semantic search architecture. One person was responsible for fine tuning the semantic search model.

Everyone contributed equally in project proposal, project status report and final project report.

References

Yoo, H., Kang, M.J., Oh, K. 2018. *A Semantic Search Model Using Word Embedding, POS Tagging, and Named Entity Recognition*. 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 1204-1209.

Bassani, E., Pasi, G.. 2021. *Semantic Query Labeling Through Synthetic Query Generation*. Proceedings Of The 44Th International ACM SIGIR Conference On Research And Development In Information Retrieval.

Aparício, M., Figueiredo, P., Raposo, F.A. 2016. *Summarization of films and documentaries based on subtitles and scripts*. Pattern Recognit. Lett., 73, 7-12.

Sanh, V., Debut, L., Chaumond, J., Wolf, T. 2019. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. ArXiv, abs/1910.01108.