ECE 1786 Creative Application of Natural Language Processing

Group: Aladdin Recommender

Group Member:

Zijian Chen (1003234582)

zijian.chen@mail.utoronto.ca

Chen Zhao (1008698358)

czchen.zhao@mail.utoronto.ca

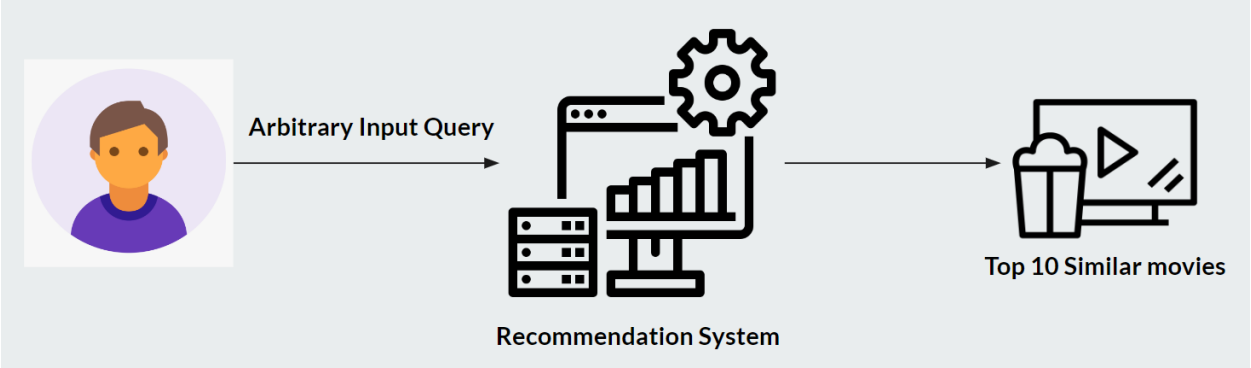
Date: 11/21/2022

Word Count: 1994 words

Introduction

As the movie industry has bloomed and developed during the past century, an increasing amount of movies are delivered and published at a fast pace. The role of movie retrieval or recommendation tools is becoming not only more important but also necessary. Even though many movie information database websites(e.g. IMDB) and video streaming service platforms(e.g. Netflix) allow users to search for the desired films/TV series, they are mostly functioned based on predefined input format such as title, genre or regions that users have no flexibility to input arbitrary plots as searching queries. The goal of our project is to construct a content-based recommendation system that accepts informal queries like: “time travel and observe the history” and suggests top movies with the most similar plots. To understand the user inputs and its semantic meaning, natural language processing techniques such as transformers are applicable to this project.

Illustration / Figure



Background & Related Work

1. Content-based Recommender Using Natural Language Processing (NLP)[1] introduced a content-based recommender that recommends movies based on the cosine-similarity of tf-idf scores in terms of user input movies. The recommender will find similar movies based on user input.
2. Transformers4Rec: A flexible library for Sequential and Session-based recommendation[2] introduced a package ‘transformers4rec’ designed by Nvidia Merlin which is a session-based recommendation aimed at capturing sequential patterns in users browsing and might help to anticipate the next user interests for better recommendation using transformers. However, since we are building a content based transformer based on given contents instead of sessions, this could be a qualified reference for our project.

Data and Data Processing

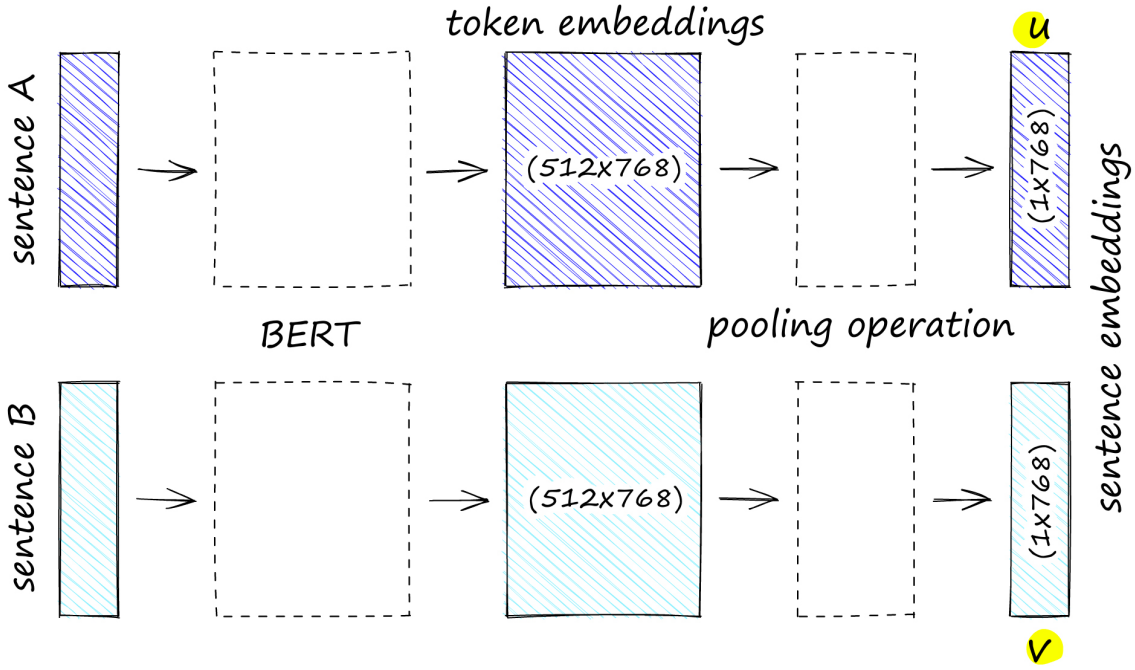
The initial dataset "IMDB5000[3]" is downloaded from kaggle, which contains around 5000 samples with features like "Title, Overview, Genre, etc". To better generalize our non-sentence features like "genre" or "release year", we construct sentences to embed these features as enriched sentences such as "The genre of the movie is action", "This movie was originally released in 2000 " to give them richer meanings instead of just words and numbers. Since we expect more comprehensive contents of movie storylines and maps the movie complete sentence (overview + storyline + enriched sentence) to reviews, the storylines from IMDB and reviews from Rotten Tomatoes are crawled and combined with initial information. Considering that data is injected from diverse resources in different formats, we establish a data pipeline to filter movies released year before 1970 which contain less reviews, clean the text format using regex to remove irrelevant symbols, select top three crews instead of all, and fetch required information from json documents. Notably, reviews are highly subjective contents from viewers. Learning from both positive reviews and negative reviews should be differentiated. Thus, classified positive and negative reviews are separated and grouped as pairs with complete sentences respectively as training data. As our model is evaluated based on its hand-labeled results, train-test splits are not necessary for our training phase and more data in the training phase reduces the data bias and variance while recommending to users.
There are 1728 final training pairs corresponding to 923 movies with 11559 appropriate reviews (6 per movie before filtering and 12 per movie after filtering) and 4504 storylines added in total.

	movie_id	title	tags	review_sentiment	sentiment_score	grouped_reviews
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	positive	1.0	Cameron's epic can still thrill the audience w...
1	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	negative	0.0	Five hundred million dollars wasted ..The leve...
2	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	positive	1.0	It still might not be quite the conclusion we ...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...	negative	0.0	Even with a twisting mystery and numerous new ...
4	49529	John Carter	John Carter is a war-weary, former military ca...	positive	1.0	John Carter is a good summer movie in March bu...
...	...	...	...	...	...	...
1723	2292	Clerks	Convenience and video store clerks Dante and R...	negative	0.0	...the films inherent deficiencies are general...
1724	14337	Primer	Friends/fledgling entrepreneurs invent a devic...	positive	1.0	Time travel may provide the paradoxical mechan...
1725	14337	Primer	Friends/fledgling entrepreneurs invent a devic...	negative	0.0	The storytelling is so confusing and the multi...
1726	126186	Shanghai Calling	When ambitious New York attorney Sam is sent t...	negative	0.0	A star is born in Daniel Henney in the predict...
1727	126186	Shanghai Calling	When ambitious New York attorney Sam is sent t...	positive	1.0	Daniel Henney is personable as Sam, and Eliza ...

1728 rows × 6 columns

Architecture and Software

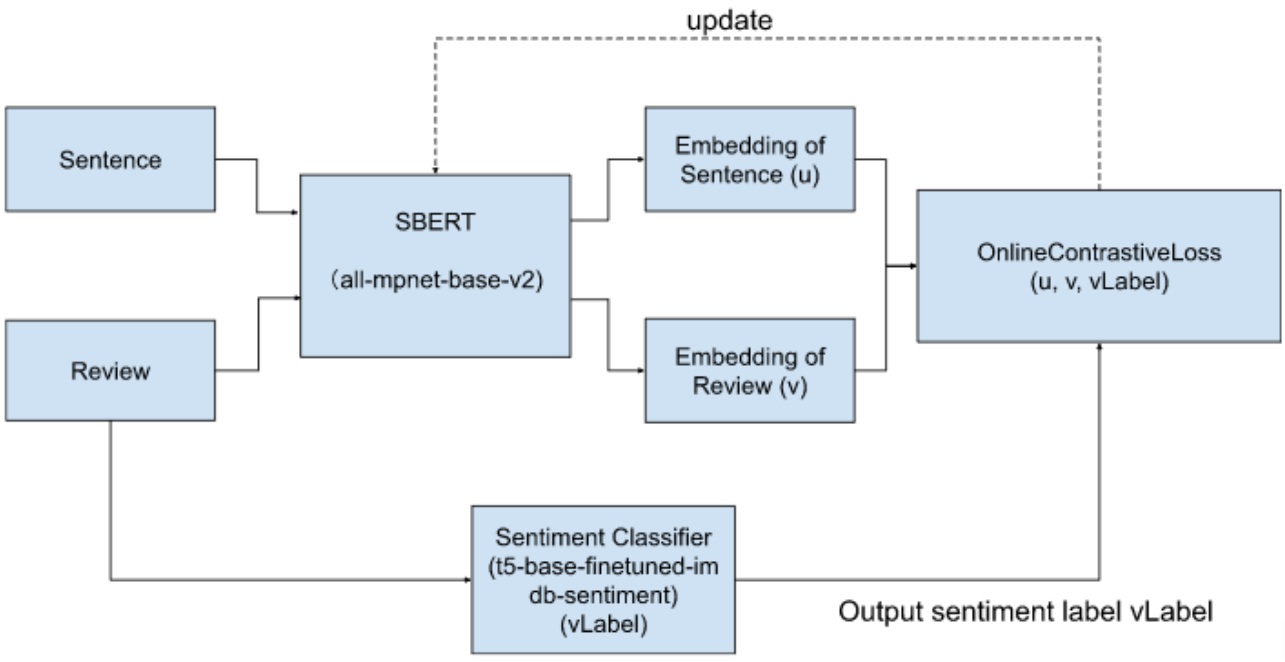
The design of our architecture will accept users arbitrary sentence inputs and return the top most relevant movies based on movies complete sentences using minimization of cosine-similarity. The complete sentences and reviews pairs are training data converted to embedding spaces using SBERT (sentence-BERT[4]) model (all-mpnet-base-v2[5]) which contains the configuration of BERT-base, with 12-layer transformer, 12 attention heads, hidden size of 768, token embedding dimension 512, and a total of 110 million parameters. The model is pre-trained MPNet on a 160 GB corpus and maps sentences to a 768 dimensional vector space. Besides, a pooling layer is applied to reduce the dimensions of the hidden layer by averaging the outputs of neuron clusters at the previous layer.



The final loss function is OnlineContrastiveLoss which will minimize the distance between the embedding space of complete sentences to positive reviews and maximize the distance between embedding space of complete sentences to negative reviews. To separate the sentiment groups, a sentiment classifier, t5-base-finetuned-imdb-sentiment[6], containing 220 millions parameters, 32128 vocabulary size, 512 token embedding dimensions, 12 layers transformer and 12 attention heads, is imported to split reviews into positive and negative groups.

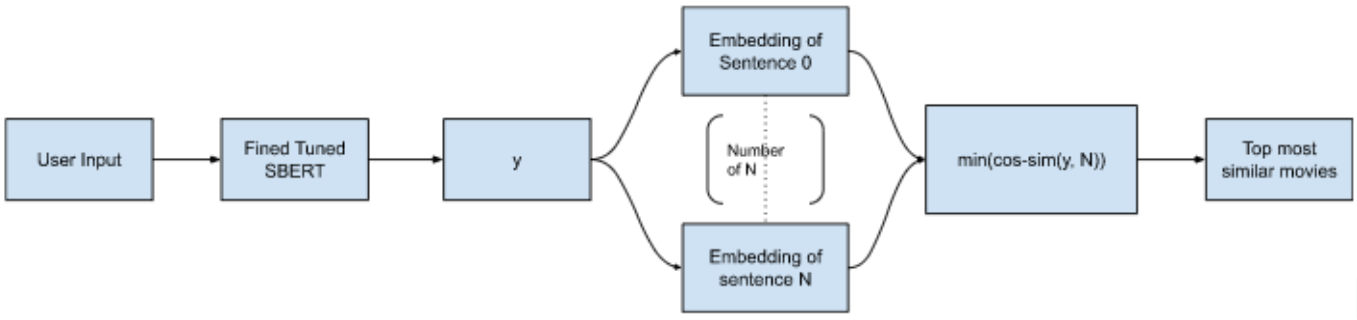
In this way, our model output will be closer to positive reviews and far away from negative results. As the training phase shown below, the OnlineContrastiveLoss is used to update our model through backpropagation and the epochs and batch size are 10 and 4 respectively with default learning rate 2e-05 due to limitation of computation power.

Training Phase

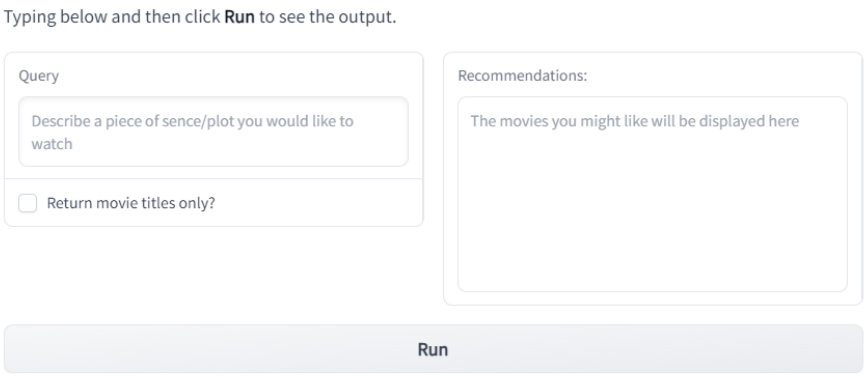


During the recommendation phase, the users can input anything as sentences which are converted into sentence embedding(y). Then the embedding(y) is compared to all other movies' complete sentences using minimization of cosine-similarity which yields the top most similar movies as our outputs.

Validation Phase

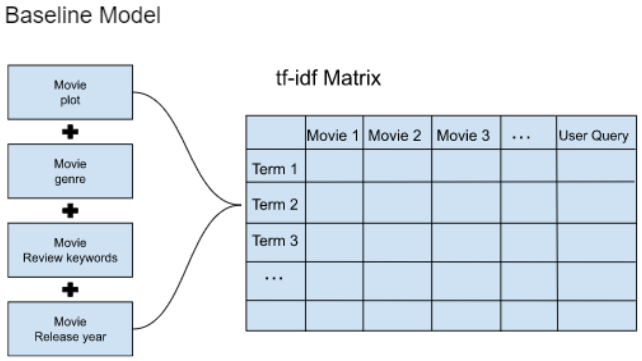


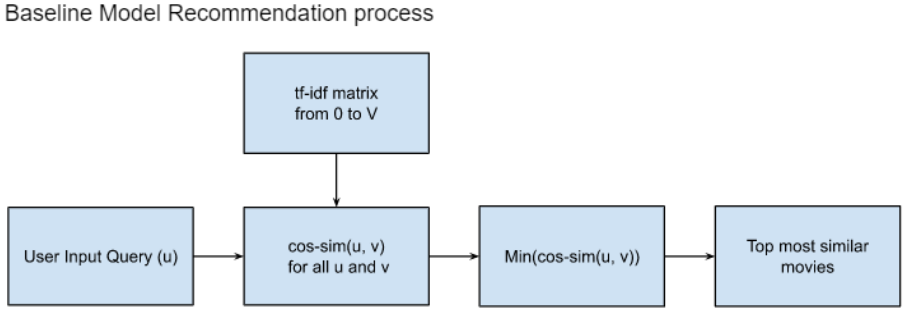
Besides, a UI interface is constructed using Gradio embedding our model to the webpage and allowing users to input queries and observe results.



Baseline Model

After data cleaning and preprocessing, the baseline model supports the same functions as transformers, which accepts users arbitrary sentence inputs and returns the top most relevant movies based on input queries. Our baseline model mainly uses tf-idf (term-frequency and inverse-document-frequency) scores to vectorize all movie information such as plot, genres, etc and form the tf-idf matrix. The recommender will iterate through the matrix of vectors and find the top most similar movies vectors given user input queries using minimization of cos-similarity. Since tf-idf vectorizers have less concern about the sentences ordering but concentrate more on tokens, all relative information could be added at tails. Besides, n-grams are used for lengths of 1 to 3 in order to expand our corpus size and extract more combinations of tokens.





Quantitative Results

To acquire quantitative results, each of our group members came up with five informal queries, then manually label each of the top 10 movie recommendations as “T”(relevant) or “F”(not relevant). Next, we exchange queries and do the labeling work again. We measure both our results separately with three different metrics and the three averages are taken as our final quantitative result.
The three highly correlated metrics we used are: Precision at K, Averaged-Precision at K and Mean Averaged-Precision at K.
“Precision at k” or “P@k” is the fraction of relevant items in the top K recommended results, it lets us know how many relevant movies among all movie recommendations can our model generate if the order of recommendations returned does not matter. Most precisions for the top ten returned movies are above 0.6 which are relatively good for single queries. Even though P@k cannot summarize the overall performance of our recommender, it better illustrates how individual queries results.

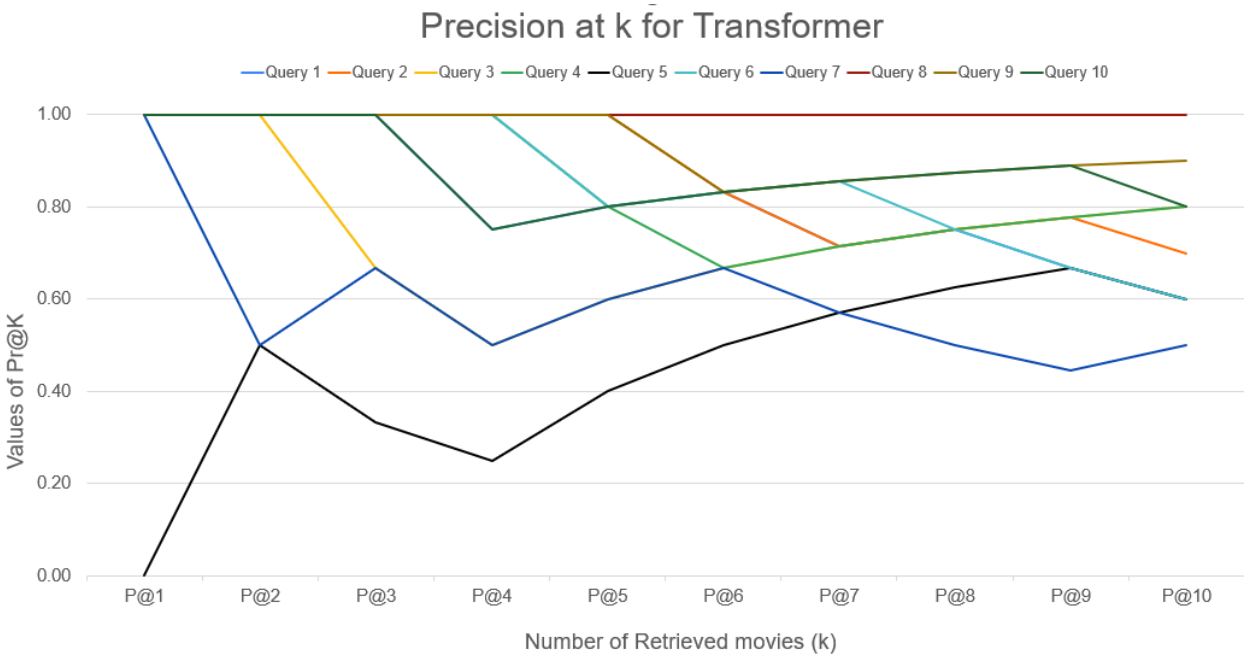


Chart and Table of “P@k” for Transformer

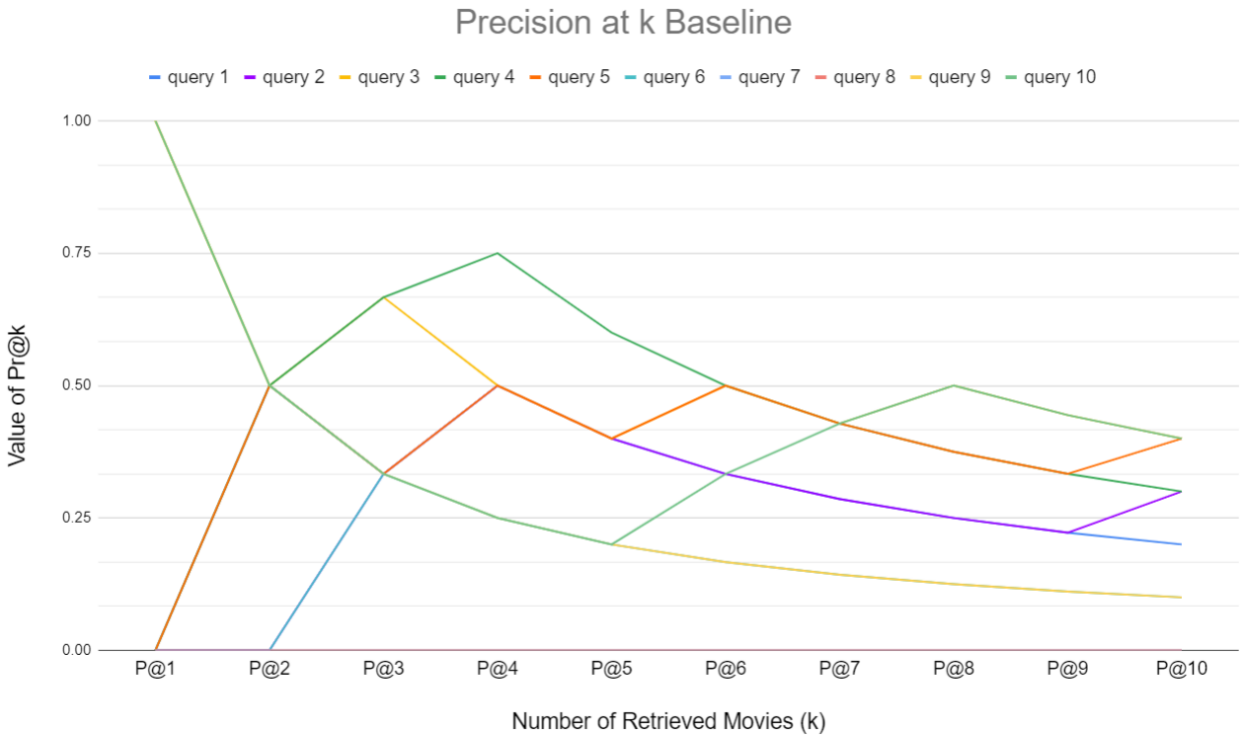


Chart and Table of "P@k" for Baseline

From above P@k values for all 10 queries, transformer model with most P@k values larger than 0.5 at k=10 generally performs better than baseline model with most P@k values smaller than 0.5 at k=10

"Average Precision at k" or "AP@k" is the sum of the Precision@K for different values of K only where the Kth recommendation is a relevant recommendation, then divide the sum by the total number of relevant items in the top K results.

$$AP@N = \frac{1}{m}\Sigma_{k=1}^{N} P(k)$$

Where m is the total number of relevant returns
P(k) = 0 if kth element is irrelevant

AP@K gives higher ranking recommendations more weights than the lower ranking ones and performs well under recommendation situations where users mostly focus on top suggested items especially for movies. We can observe this characteristic from the change of AP@K values for Transformer in the chart below. Query 10(dark green) has the most relevant returns at the head and irrelevant returns at the tail, which yields a high overall AP@k sores. However, Query 5(black), having the most relevant returns at the tail but irrelevant returns at head, results in an overall low AP@k score. Comparing the baseline with the transformer model, the overall

performance of the transformer model yields higher AP@k scores.

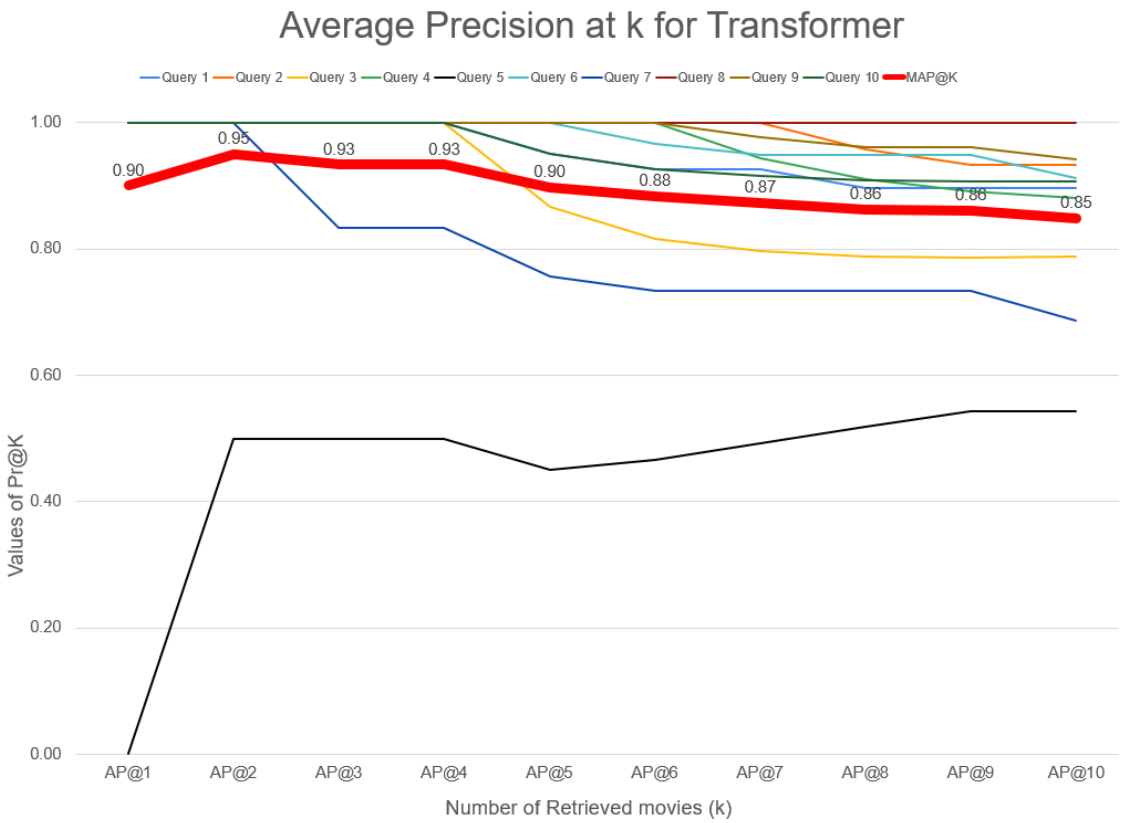


Chart and Table of "AP@k" for Transformer

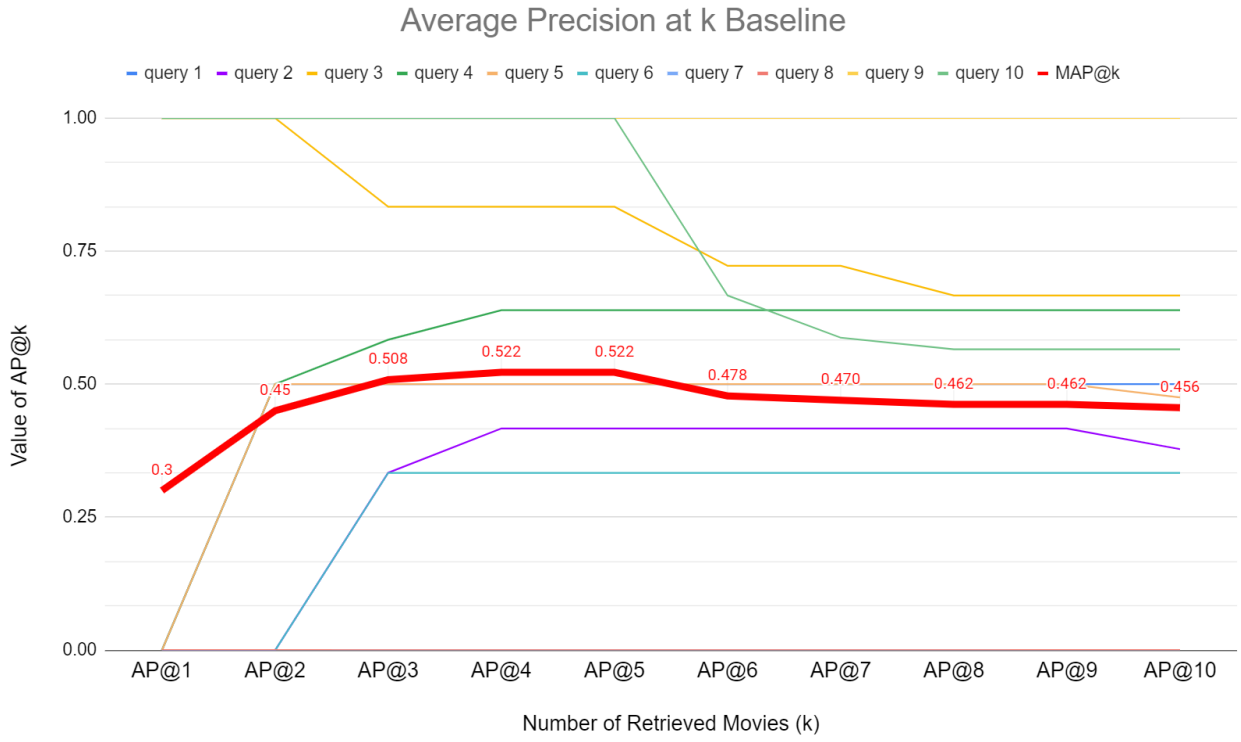


Chart and Table of “AP@k” for Baseline

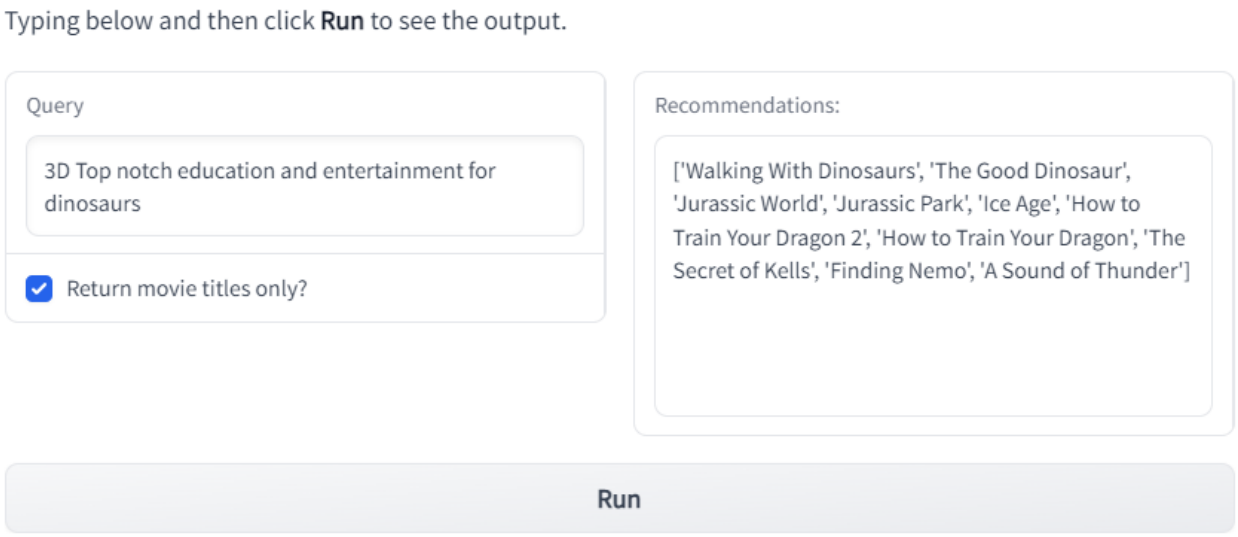
“Mean Averaged-Precision@k” or “MAP@k” is the mean of the average precision at K metric across all instances in the dataset. P@k and AP@k are used to evaluate the performance of a single query, while MAP@k evaluates the general performance of the recommendation system for all queries.

With MAP@K shown as the thick red line in the above Transformer chart with data values added on lines, our model achieves a MAP always above 0.85 for all queries. However, the MAP values for the baseline model fluctuate between 0.5. The higher the MAP score, the more accurate the model is in its suggestion. Given the trend of MAP line of transformer plot, most of our recommendations are relevant and the top five recommendations fit the user queries really well.

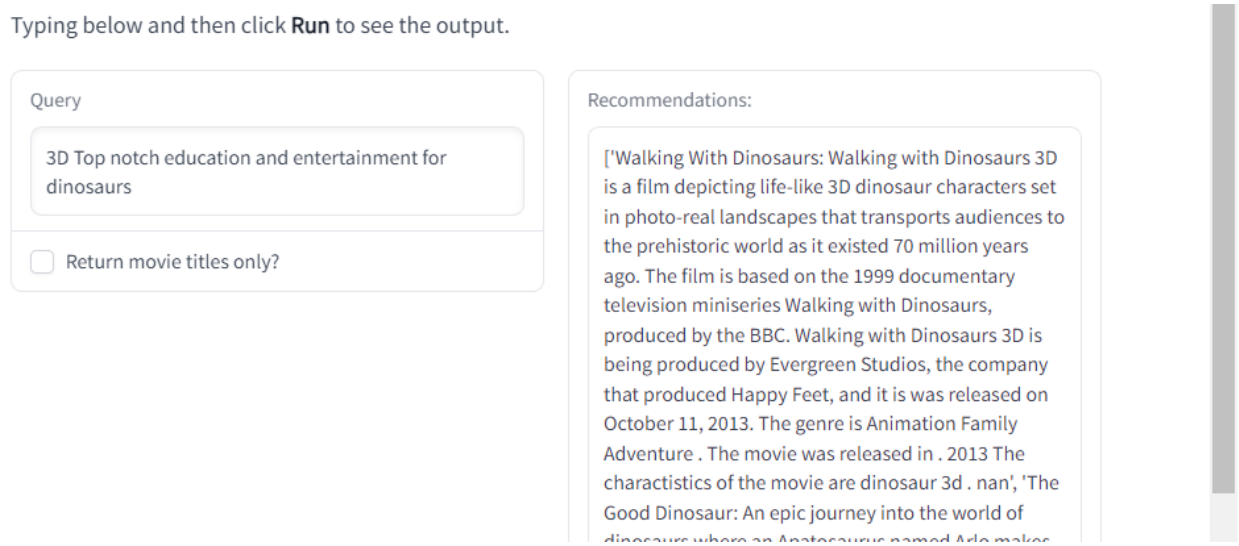
Qualitative Results

The qualitative result of our model is shown in the figure below: we created a simple user interface that allows users to input any descriptions, the top 10 recommendations will be displayed. Users also have the choice of asking for more information such as a short description, the movie’s genre, etc.

The below query results are pretty good. Intuitively, most movies are directly related to dinosaurs and even if the movies do not include dinosaur elements, they are still relevant to other characteristics in the query such as education, 3D or entertainment.

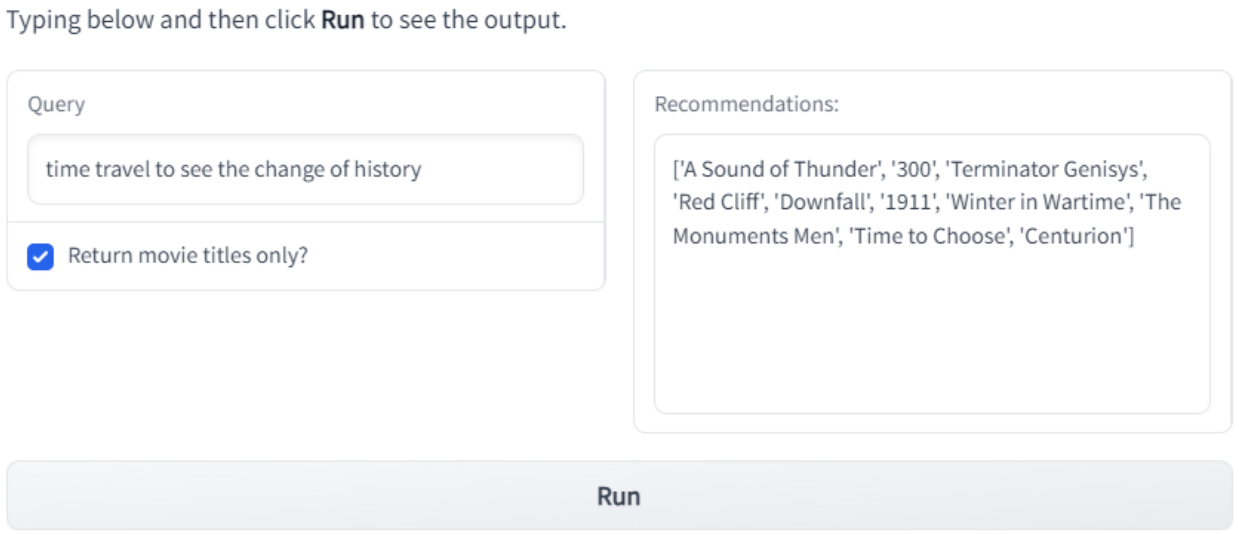


Title-only result



Result with full information

The below query results are ambiguous and hard to define its relevance. Most movies are directly related to history changes. But considering its completeness, only the first results are relevant to time travel.



Example of ambiguous results

Discussion and Learnings

The model performance is surprisingly well based on the results from both quantitative and qualitative results, especially the high values of MAP. One thing we noticed during the development of our model is that because of how informal queries have vague meanings compared to solid-feature queries, people could interpret them differently. This means no matter

how accurately our model performs, personalized preference setting should be considered for users. Even if this is a common situation for a pure content-based recommendation system without considering user preferences, it could be certainly improved by combining the session-based or collaborative-based methods if we start again by shifting some concentration from NLP applications to results improvements and availability out of this course. Besides, a more comprehensive dataset would be much useful that more movies can be trained on instead of filtered out without enough information The best usage scenario for our model could be embedded into the search engine of a movie website, we are allowed to not only apply some hard constraint to the searching result(e.g. Language, R-rating), but also adjust the model based on the user's choice of recommendations.

Individual Contributions

Zijian

1. Implemented the web-scraping codes to acquire data from rotten tomatoes, IMDB
2. Crawled data from rotten tomatoes
3. Clean and preprocess sources data to combine results
4. Experimented with various methods to improve model performance
5. Fine-tuned transformer models with various loss functions and model structure
6. Constructed enriched sentence to complete sentences
7. Hand-labeled both baseline and transformer results
8. Generate output plots
9. Wrote the Gradio implementation of the user-facing side of the project
10. Report writing

Chen

1. Crawled data from imdb
2. Clean and preprocess sources data to combine results
3. Implemented the baseline models and transformer models
4. Experimented with various methods to improve model performance
5. Fine-tuned transformer models with various loss functions and model structure
6. Imported t5 sentiment model and tried OnlineContractiveLoss
7. Hand-labeled transformer results
8. Researched on available metrics
9. Modified output plots
10. Report writing

Reference

1. J. Ng, “Content-based recommender using Natural Language Processing (NLP),” *KDnuggets*. [Online]. Available: https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html.
2. G. Moreira, “Transformers4Rec: A flexible library for sequential and session-based recommendation,” *Medium*, 08-Nov-2021. [Online]. Available: https://medium.com/nvidia-merlin/transformers4rec-4523cc7d8fa8
3. T. M. D. (TMDb), “TMDB 5000 movie dataset,” *Kaggle*, 28-Sep-2017. [Online]. Available: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
4. N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using Siamese Bert-Networks,” *arXiv.org*, 27-Aug-2019. [Online]. Available: https://arxiv.org/abs/1908.10084
5. O. Espejel, “Sentence-transformers/all-mpnet-base-V2 · hugging face,” *sentence-transformers/all-mpnet-base-v2 · Hugging Face*. [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2.
6. M. Romero, “MRM8488/T5-base-finetuned-imdb-sentiment · hugging face,” *mrm8488/t5-base-finetuned-imdb-sentiment · Hugging Face*. [Online]. Available: https://huggingface.co/mrm8488/t5-base-finetuned-imdb-sentiment

Appendix

Result of Baseline model

p@k	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
query 1	0	0.5	0.33333	0.5	0.4	0.3333333333	0.2857142857	0.25	0.2222222222	0.2
query 2	0	0	0.33333	0.5	0.4	0.3333333333	0.2857142857	0.25	0.2222222222	0.3
query 3	1	0.5	0.66666	0.5	0.4	0.5	0.4285714286	0.5	0.4444444444	0.4
query 4	0	0.5	0.66666	0.75	0.6	0.5	0.4285714286	0.375	0.3333333333	0.3
query 5	0	0.5	0.33333	0.5	0.4	0.5	0.4285714286	0.375	0.3333333333	0.4
query 6	0	0	0.33333	0.25	0.2	0.1666666667	0.1428571429	0.125	0.1111111111	0.1
query 7	0	0	0	0	0	0	0	0	0	0
query 8	0	0	0	0	0	0	0	0	0	0
query 9	1	0.5	0.33333	0.25	0.2	0.1666666667	0.1428571429	0.125	0.1111111111	0.1
query 10	1	0.5	0.33333	0.25	0.2	0.3333333333	0.4285714286	0.5	0.4444444444	0.4

ap@k	AP@1	AP@2	AP@3	AP@4	AP@5	AP@6	AP@7	AP@8	AP@9	AP@10
query 1	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
query 2	0	0	0.33333	0.4166666667	0.4166666667	0.4166666667	0.4166666667	0.4166666667	0.4166666667	0.3777777778
query 3	1	1	0.83333	0.8333333333	0.8333333333	0.7222222222	0.7222222222	0.6666666667	0.6666666667	0.6666666667
query 4	0	0.5	0.58333	0.6388888889	0.6388888889	0.6388888889	0.6388888889	0.6388888889	0.6388888889	0.6388888889
query 5	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.475
query 6	0	0	0.33333	0.3333333333	0.3333333333	0.3333333333	0.3333333333	0.3333333333	0.3333333333	0.3333333333
query 7	0	0	0	0	0	0	0	0	0	0
query 8	0	0	0	0	0	0	0	0	0	0
query 9	1	1	1	1	1	1	1	1	1	1
query 10	1	1	1	1	1	0.6666666667	0.5873015873	0.5654761905	0.5654761905	0.5654761905
MAP@k	0.3	0.45	0.50833	0.5222222222	0.5222222222	0.4777777778	0.4698412698	0.4621031746	0.4621031746	0.4557142857

Result of Transformer model

P@k	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10
Query 1	1	1	1	0.75	0.8	0.833333333	0.714285714	0.75	0.666666667	0.6
Query 2	1	1	1	1	1	0.833333333	0.714285714	0.75	0.777777778	0.7
Query 3	1	1	0.666666667	0.5	0.6	0.666666667	0.714285714	0.75	0.777777778	0.8
Query 4	1	1	1	1	0.8	0.666666667	0.714285714	0.75	0.777777778	0.8
Query 5	0	0.5	0.333333333	0.25	0.4	0.5	0.571428571	0.625	0.666666667	0.6
Query 6	1	1	1	1	0.8	0.833333333	0.857142857	0.75	0.666666667	0.6
Query 7	1	0.5	0.666666667	0.5	0.6	0.666666667	0.571428571	0.5	0.444444444	0.5
Query 8	1	1	1	1	1	1	1	1	1	1
Query 9	1	1	1	1	1	0.833333333	0.857142857	0.875	0.888888889	0.9
Query 10	1	1	1	0.75	0.8	0.833333333	0.857142857	0.875	0.888888889	0.8

AP@k	AP@1	AP@2	AP@3	AP@4	AP@5	AP@6	AP@7	AP@8	AP@9	AP@10
Query 1	1	1	1	1	0.95	0.926666667	0.926666667	0.897222222	0.897222222	0.897222222
Query 2	1	1	1	1	1	1	1	0.958333333	0.932539683	0.932539683
Query 3	1	1	1	1	0.866666667	0.816666667	0.796190476	0.788492063	0.786961451	0.78859127
Query 4	1	1	1	1	1	1	0.942857143	0.910714286	0.891723356	0.880257937
Query 5	0	0.5	0.5	0.5	0.45	0.466666667	0.492857143	0.519285714	0.543849206	0.543849206
Query 6	1	1	1	1	1	0.966666667	0.948412698	0.948412698	0.948412698	0.91292517
Query 7	1	1	0.833333333	0.833333333	0.755555556	0.733333333	0.733333333	0.733333333	0.733333333	0.686666667
Query 8	1	1	1	1	1	1	1	1	1	1
Query 9	1	1	1	1	1	1	0.976190476	0.961734694	0.961734694	0.941517857
Query 10	1	1	1	1	0.95	0.926666667	0.915079365	0.909353741	0.906795635	0.906795635
MAP@K	0.9	0.95	0.933333333	0.933333333	0.897222222	0.883666667	0.87315873	0.862688209	0.860257228	0.849036565

Permissions

Zijian Chen:
permission to post video: Yes
permission to post final report: Yes
permission to post source code: Yes

Chen Zhao:
permission to post video: Yes
permission to post final report: Yes
permission to post source code: Yes