BERT-Based Movie Keyword Search Leveraging User-Generated Movie Rankings and Reviews

Tensho Miyashita
Graduate School of Science and Engineering
Aoyama Gakuin University
Kanagawa, Japan
tensho@sw.it.aoyama.ac.jp

Yoshiyuki Shoji
Faculty of Informatics
Shizuoka University
Shizuoka, Japan
shojiy@inf.shizuoka.ac.jp

Sumio Fujita
LY Corporation
Tokyo, Japan
sufujita@lycorp.co.jp

Martin J. Dürst
College of Science and Engineering
Aoyama Gakuin University)
Kanagawa, Japan
duerst@it.aoyama.ac.jp

***Abstract*—This paper introduces a novel method for movie keyword searches based on user-generated rankings and reviews. We utilize the capabilities of the BERT language model, which has been enriched with task-specific fine-tuning. The model is trained to understand the relationship between keywords and movies using paired user-generated ranking titles and movie reviews. We sourced our data from a renowned Japanese movie review platform. This dataset comprises 10,000 user rankings and 15,000 films. In a binary classification task, our model demonstrated superior performance compared to traditional similarity-based methods. While our approach outperforms traditional similarity methods, further improvements in pooling techniques are necessary.**

***Index Terms*—LLM, Movie Search, User Generated Contents, Review, Ranking**

I. Introduction

In recent years, how people select movies has changed due to the widespread use of on-demand and subscription-based movie distribution services. Conventionally, a person goes to a movie theater and selects a movie to watch on the spot. In this way, since only a few movies are played in theaters, they only have to choose one movie to watch from several candidates. Even when using video rental stores, the number of videos is limited; therefore, users can decide on a movie to watch before coming to the store or search for a movie to watch in store. However, when using movie distribution sites, users need to select a movie from almost all movies ever released. In this case, it is difficult to examine each movie individually to see if it suits their demands.

Attitudes toward watching movies are also changing. Recently, more and more people have been casually watching movies on their computers and smartphones. Watching movies while working and increasing playback speed is becoming more common; watching movies is becoming more integrated into everyday life. This change in movie-watching style has increased the demand for technology that enables users to search for movies more easily.

For example, there is a need to search for movies using the ranking of arbitrary keywords, such as “I want to see movies with beautiful scenery that I can play while working” or “I just want to see movies with flashy action.” However, conventional movie searches cannot find movies using keywords such as “beautiful scenery” and “flashy action” because movie information sites target descriptions and metadata. Metadata on movie information websites generally only include information such as the original title, country of production, director, music, actors, *etc.*, as well as descriptions and synopses. However, the description or synopsis of a movie is merely a text that describes what the movie is about from a third party’s point of view. It does not include information about the impressions the movie made on people who watched it.

In this situation, reviews are an important source of information for judging how viewers feel about a movie, such as whether the movie has “beautiful scenery.” However, when trying to find movies with “beautiful scenery” from the vast number of movies available on video distribution sites, it is not realistic to read the reviews posted for each movie one by one to determine whether it is a movie that the user wants to see.

In this paper, we propose a method for ranking movies based on free queries by learning the relevance of movies and queries from user-generated rankings and movie reviews using bidirectional encoder representations from transformers (BERT). For this purpose, we focused on services that allow users to submit their own movie rankings. Some movie review sites have a function in which individuals can create a list of their favorite movies. For example, Yahoo! Movies, one of the biggest movie information sites in Japan, uses Round-Up to provide this function, where users can register up to 10 movies using any point of view they like. Each user-generated ranking in Yahoo! Movies’ Round-Up is given a title, for example, “Top 10 movies that make me cry,” and then 10 movies are listed.

We hypothesized that these ranking titles might function like search keywords. Therefore, we attempted to make movies searchable using keywords by learning the relationship between ranking titles and the movies listed in the ranking. In recent years, inference using large-scale language models has become more common, as is typified by BERT. If such a language model can infer that "movies with this type of review tend to appear in rankings with this kind of title," then movies can be searched for using arbitrary keywords. In other words, movies can be scored by the probability of their appearance in user-generated rankings whose titles contain the keyword query.

To achieve such a search method, we proposed an algorithm comprising three steps (see Figure 1):

- Vectorize ranking titles and all reviews using BERT,
- train the neural network model to calculate the relevance between the ranking title and reviews of a certain movie, and
- generate rankings from a user-entered query.

The first step was the vectorization of keywords and reviews. We used BERT, one of the most popular large language models, to vectorize sentences consisting of natural language, such as queries and reviews. The feature vector archived by BERT is a numerical representation of the meaning of a sentence or word.

The second step involved learning the relevance of the queries and reviews using machine learning. We used a simple neural network model and trained it with movie vectors and ranked title vectors. The user-generated rankings were used for this learning as the training data. For example, suppose a certain movie appeared in a ranking named "My top 10 tearjerker movies." The model would then learn the movie review and the ranking title as a pair. Such learning enables the model to solve binary classification tasks for pairs of keywords and reviews. For example, whether a movie with a particular review appears in a user-generated ranking that includes the keyword in the title.

The third step was to generate actual rankings from user-entered queries. Here, the keyword query was vectorized using BERT. The algorithm combined the vector of queries with the vector of movies and then classified them using the trained model. That is, whether the movie is likely to appear in the ranking with the query in its title. The algorithm can rank movies according to their classification probability by performing a brute-force classification of all candidate movies. Here, we expected the performance of BERT's zero-shot inference to be high enough to rank movies even by keywords that were not included in the ranking title.

In this way, generating a movie ranking based on arbitrary keywords was possible. We conducted a subject experiment to confirm the accuracy and effectiveness of this ranking. Participants were asked to evaluate how closely the movies in the rankings generated by the proposed method and several comparison methods were related to the query.

This paper is an extended version of the short paper previously published [1].

II. Related Work

This research aimed to make movies searchable using word of mouth (eWOM). BERT and Learning to Rank were used as enabling technologies.

A. Item Search Using eWOM

Our research used reviews posted on movie sites, a type of eWOM, to find movies that were close to what users were looking for. There are several examples of this kind of item search that focus on eWOM.

Ramanand *et al.* [2] proposed a method for extracting "wishes" that indicated suggestions about products and services and purchase intentions from documents, such as reviews and buyer surveys. The "wishes" extracted from the reviews were applied not only to improve the quality of products and services but also the recommendations and presentations of products sought by customers. Similarly, this study used user review information to search for movies using free queries.

B. Information Retrieval Using BERT

BERT is a natural language processing model proposed by Devlin *et al.* [3] that enables context reading. BERT has been applied in various fields because it can convert the meaning of words in a sentence into a distributed representation of vectors with high accuracy.

Depending on the application, BERT has been used for various task-specific purposes by fine-tuning pretrained language models. As an example of fine-tuning focusing on sentence similarity, Nils *et al.* [4] proposed Sentence Bert, a BERT-based model specialized for sentence similarity searches. Experiments evaluating the performance of sentence embedding using SentEval have shown that this method can vectorize the meanings of sentences more effectively.

As another example of a specific domain, Zhuang *et al.* [5] proposed a fine-tuning BERT model that focuses on words related to finance, FinBERT. Using performance evaluation experiments using FiQA, a dataset consisting of question-and-answer documents in the field of finance showed that FinBERT effectively reflects the meanings of finance-related words in vectors. Shibata *et al.* [6] proposed a method for retrieving query-relevant FAQs using BERT. Experiments using the localgovFAQ and StackExchange datasets showed that this method enabled more accurate FAQ retrieval. This study used simple BERT without fine-tuning in the specific domain of movie reviews.

There are also examples of BERT applied to information retrieval. Yang *et al.* [7] proposed a method for adapting BERT to the ad hoc retrieval of documents. Experiments with TREC Microblog Tracks showed that BERT-based methods are effective for retrieving short documents, such as those found in microblogs. Yunqiu *et al.* [8] proposed a BERT model for legal case retrieval, BERT-PLI, that can retrieve from much longer queries than general queries. A related precedent retrieval task using the COLIEE 2019 dataset revealed that this method can more accurately understand the meaning of longer documents. Zhuolin *et al.* [9] proposed a

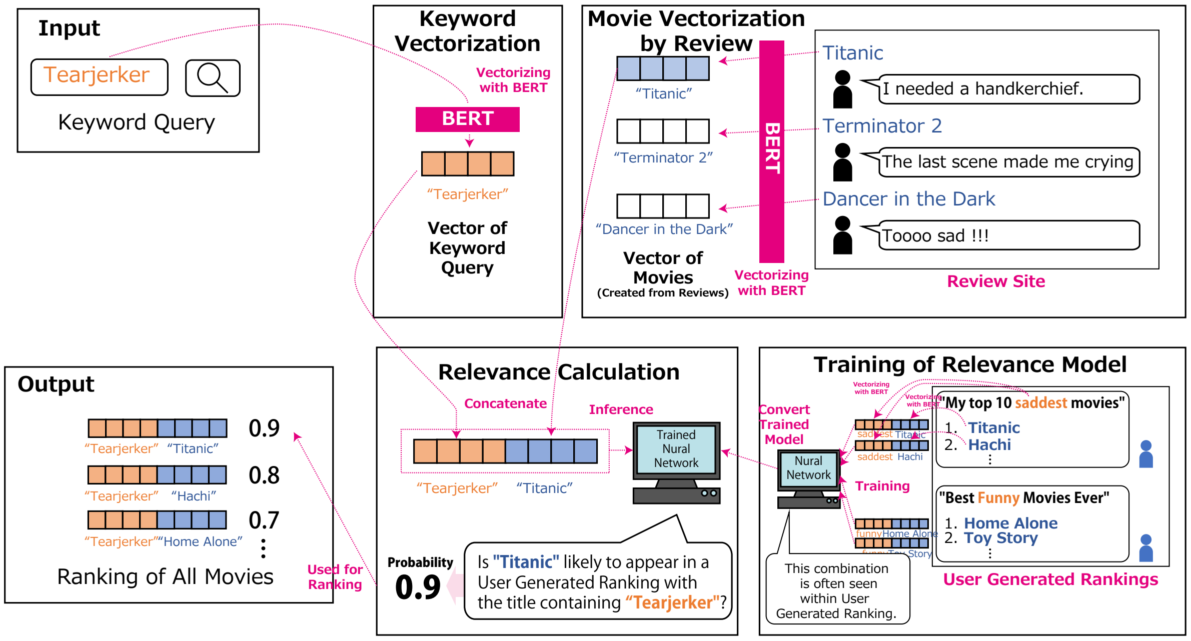

Fig. 1. Overview of our proposed method. Estimating the relevance between the query and movies using the user-generated ranking as the correct answer. When the model inputs a vector of queries and an arbitrary movie, it outputs the probability that the combination exists in the correct answer dataset.

method for retrieving multilingual documents from English queries by using BERT to learn the relationships between English queries and multilingual documents. The task of retrieving Lithuanian text and audio documents using the query and search corpus provided by IARPA's MATERIAL program revealed that this method enables more accurate retrieval than other methods.

Since this research was conducted on movies, which are visual images, it is difficult to deal with the contents of movies in text; therefore, we used user-posted reviews. In addition, it is difficult to compare the similarity of review sentences and short queries because of the different natures of the sentences. Therefore, we used Learning to Rank to match queries and review sentences.

C. Learning to Rank

Learning to Rank is a method for solving ranking problems using machine learning. It is also called ranking learning. Three main approaches to Learning to Rank [10] are pointwise, pairwise, and listwise methods. In this study, we used the pointwise method. The Learning to Rank method aims to rank highly relevant documents in response to a query, and it is mainly used for information retrieval.

As an example of retrieving documents belonging to a specific topic, Amir *et al.* [11] proposed a method that used BERT and Learning to Rank to retrieve evidence to support a claim. The task of retrieving sentences that provided evidence using the FEVER dataset showed that this method performed best compared to other methods.

Yu *et al.* [12] proposed a method using Learning to Rank to find documents containing answers to a given question. Experiments on a standard TREC benchmark dataset showed that the method's performance was comparable to state-of-the-art methods used to retrieve answer sentences. Fabio *et al.* [13] proposed a method using Learning to Rank for content-based image retrieval. Evaluation experiments on two datasets, Corel GALLERY Magic Stock Photo Library 2, and Caltech, showed that the method outperformed traditional ranking methods. Since the current research is concerned with the retrieval of an item (*i.e.,* a movie), we also discuss some examples of applying Learning to Rank to item retrieval.

Shubhra *et al.* [14] proposed a method that applied Learning to Rank for product retrieval on an e-commerce site. They found that the method, which used LambdaMART, a ranking learning method, provided the highest score for a search task using randomly selected queries and product information from the web. Blake *et al.* [15] proposed a method using Learning to Rank to improve the accuracy of location retrieval using inaccurate GPS data. Experiments using actual check-in information about facilities in Manhattan showed that this proposed method had the highest accuracy for locating locations.

Prior to this study, Kurihara *et al.* [16] proposed a method using Learning to Rank for movie retrieval based on review information. They proposed a model that estimated the ranking order of movies using a decision tree-based method called LambdaMART. However, the current research aimed to make it possible to rank movies in a new way by learning the relevance of queries and items using a neural network.

III. Method

In this study, we propose a search algorithm that ranks movies in response to an arbitrary keyword query. An overview of the method is shown in Figure 1. The method first extracts the titles and movies included in the rankings from user-generated movie rankings. Next, both the words and movies in the ranking titles are vectorized using BERT. The two vectors are then concatenated and input into neural networks, which learn them as a binary classification problem. That is, does a movie appear in the rankings with this keyword in its title?

By inputting an arbitrary query and movie into the learned model, it is possible to estimate how likely the movie will appear in the rankings that include the query as a title. By calculating the probability of appearance for all movies, a movie ranking can be generated from an arbitrary query.

A. Vectorizing Movies Using Reviews

Movie websites contain movie reviews posted by various users. In this study, it was assumed that user-posted reviews contain information that represents the movie's features. Therefore, when representing a movie as a vector, a review sentence for that movie was used instead of a movie's metadata or visual information.

First, the text was preprocessed to vectorize the review sentences. Since there was a limit to the number of tokens that BERT could vectorize, it was impossible to vectorize long review sentences. Therefore, the review sentences were split by punctuation marks, question marks, exclamation marks, and other symbols. Unnecessary characters and symbols were also removed.

Next, each preprocessed reviewed sentence was input into BERT to calculate a 768-dimensional vector in a distributed representation format. However, BERT could not convert long sentences into distributed representations due to the upper limit of computational complexity. Therefore, each sentence $r \in R(m)$ in the movie review m was vectorized and treated as a feature vector of the movie by pooling it. When the vector representing any review sentence r was denoted by $BERT_r(r)$, the vector $\mathbf{v}(m)$ of movies m was defined as follows:

$$\mathbf{v}(m) = \frac{\sum_{r \in R(m)} \mathrm{BERT}_r(r)}{|R(m)|}. \quad (1)$$

This is the average of the vectors of all the review sentences for that movie.

B. Formatting User-Generated Rankings as Training Data

To enable movie retrieval for a specific query, correct training data linking the query to the movies was required. Therefore, this study used data from websites where users can submit their own movie rankings as the correct data. We collected user-generated rankings registered in Yahoo! Movies Japan for our experiment.

In recent years, sharing arbitrary movie rankings created by individual users on the Internet has become common. For example, it is easy to find rankings such as "My personal top 10 sad movies" on personal blogs. There are also many web services that enable users to post such rankings. For example, IMDB, the world's largest movie review site, provides a function called Watchlist[1]." This feature enables users to create a list of arbitrary titles and register their favorite movies on the list. In Japan, Yahoo! Movies' Round-Up enables users to create a list with an arbitrary title and register up to 10 movies. Many of these individually created rankings include titles such as "The 10 best movies that make me cry" or "The 10 best comedy movies I like." In this study, the model learned the relationship between queries and movies by using these kinds of user-generated rankings as correct answer data.

The data were not clean because the general public creates these user-generated rankings. They contain many irrelevant descriptions and miscellaneous rankings and therefore require preprocessing. First, we excluded rankings that were not based on a specific point of view. These rankings vary widely; some users simply rank the movies they like. For example, rankings such as "10 Favorite Movies" or "My Top 10 Movies of 2002" do not contain a specific point of view. We manually created dictionaries to eliminate such rankings. We eliminated rankings that included years or specific words, such as "all-time." Words that were unlikely to be connected to the query were then removed from the ranking titles. Specifically, these were words such as "my," "top," and "movie," and adverbs.

Next, the training data were formatted to train the model. The training data consisted of input data and a correct answer label. The title of the ranking and the movies in it were vectorized using BERT. Each pair of movie and ranking title vectors was combined and used as input to the neural network. The correct answer label was a binary value of 0 or 1, indicating whether the movie was included in the ranking. Positive examples were generated using the movies included in the ranking. Negative examples were randomly generated from movies that were not included in the ranking.

C. Learning the Relationship Between Movies and Terms in the Ranking Title

Relevance between a query and a movie is computable by considering the ranking title as a keyword query. Here, a simple neural network was used to calculate relevance. When each movie and query has been represented as a vector of distributed representations, the proposed method trains a neural network as a binary classification task using these vectors, as shown in Figure2. The task is to combine vectors of movies and queries and then estimate whether the movie will likely appear in a ranking that includes the query in its title.

The input was a 1,536-dimensional vector: a combined pair of a 768-dimensional vector representing a movie and a 768-dimensional vector representing a ranking title. This neural network consisted of four full-combination layers, as shown in Table I. The output layer was binary since it performs a binary classification of whether a movie appears in the ranking whose title contains a given query.

[1]IMDB Watchlist: https://imdb.com/list/watchlist

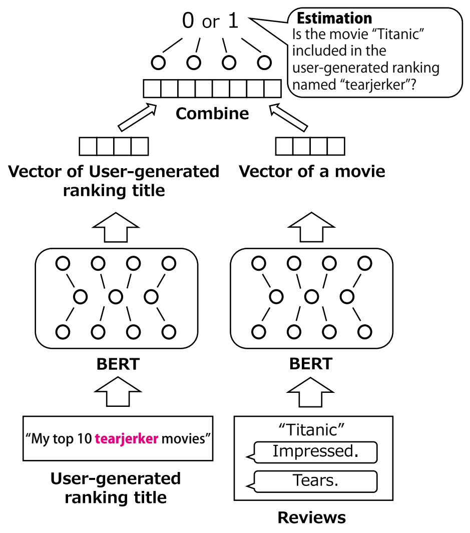
0 or 1
Estimation
Is the movie "Titanic" included in the user-generated ranking named "tearjerker"?
Combine
Vector of User-generated ranking title
Vector of a movie
BERT
BERT
"My top 10 tearjerker movies"
User-generated ranking title
"Titanic"
Impressed.
Tears.
Reviews

Fig. 2. Procedure for training the model to predict whether or not a movie will appear in user-generated rankings that include the query in the title.

TABLE I
DETAILS OF THE NETWORK LAYER USED FOR THE TRAINING.

Layer Type	# Nodes	Activation Function
fully connected	1,536	Relu
fully connected	64	Relu
fully connected	64	Relu
fully connected	2	Sigmoid

The loss function $\mathrm{loss}(p, q)$ is expressed as

$$\mathrm{loss}(p, q) = -\sum_{x} p(x)log(q(x)), \quad (2)$$

where p is the value of the correct answer, and q is the predicted value. The values obtained by this loss function were used to optimize the network model.

D. Generate Ranking for a Given Query

Using the language model learned this way, movies were ranked based on arbitrary queries. The current method calculated the relevance of a query for all movies in a round-robin fashion, and it ranked movies in order of relevance.

The incoming query was vectorized using BERT. The vector was then combined with the vector of a certain movie. These combined vectors were input into the trained model created in section III-C. This calculated the relevance of the query to the movie. This process was then repeated for all movies in a brute-force manner. Since the relevance to all movies was calculated, it could rank the movies for the query.

IV. EVALUATION

A participant experiment was conducted to verify whether the movie ranking generated by the proposed method was consistent with the user's perception. For a particular query, we listed the movies that appeared in the rankings generated by each method and asked the participants to evaluate how well they matched the query.

A. Compared Methods

Two hypotheses that can be linked to the technical contribution of this study are as follows:

H1 : Review information can be used to search for movies by keywords that do not appear in the movie synopsis or metadata.

H2 : If each of the documents and queries about a movie is represented as a vector, a simple similarity comparison, such as cosine similarity, is not sufficient.

To test these hypotheses, we compared three variant methods and the proposed methods, as follows:

- **Proposed Method**: A method for matching movie vectors generated from reviews with queries using deep learning.
- **Movie Similarity**: A method for comparing the similarity between the movie vector generated from the reviews and the query vector.
- **Review Sentence Similarity**: A method for comparing the similarity between the review sentence vector and the query vector.
- **Metadata Only**: A method using vectors of the movie's metadata, such as description text on movie sites.

Proposed Method uses deep learning to estimate the relevance of the query vector to the vectors generated from the movie reviews described in Section III. The method reflects both hypotheses H1 and H2.

Movie Similarity compares the vectors generated from the movie reviews and the query vectors using cosine similarity. It takes a query as input and vectorizes the query using BERT. Then, the cosine similarity between the query vector and the movie vector is calculated. Based on the results, movies with the highest similarity were output as a ranking. This method reflects only hypothesis H1.

Review Sentence Similarity compares the similarity between the review vectors used to generate the movie and query vectors. This calculates the cosine similarity between the vector of input queries and the vector of review sentences posted on Yahoo! Movies with highly similar reviews are ranked based on the similarity between the review text and the query. When multiple reviews posted for the same movie appear in the ranking, the similarity of the most similar review is treated as the movie's similarity. This method reflects hypothesis H1. However, hypothesis H2 is based on the assumption that superficial similarity is insufficient for comparing a query to a movie. Thus, it corresponds to a review sentence to a query instead of reaching a movie to a query.

Metadata Only uses only metadata and no reviews. The information about a movie includes metadata, such as the original title, production year, running time, production country, genre, director, executive producer, script, music, *etc.*, and text such as commentary, synopsis, *etc.* First, text matching is performed between the input query and the above information.

TABLE II
QUERIES USED IN THE EXPERIMENT AND THEIR FEATURES

Query	Features
Tearjerker Laughable Shocking	Emotion after watching the movie
Suitable for dating Suitable for children	Situation or scene when watching the movie
Suspense Animation	Category of the movie
Ghibli Surprise ending Takeshi Kitano	Content of the movie

If there are any matched movies, the cosine similarity between the query vector and the vectors of the commentary and synopsis sentences of those movies is calculated. Movies with high similarity are ranked in the output. This method does not support either hypothesis H1 or H2.

B. Dataset

We collected about 15,000 movies with 10 or more reviews from a certain real movie review site (anonimized). We collected about 40,000 movie reviews and about 10,000 user-generated rankings from Round-Up, Yahoo! Movies as correct answers for training. We used about 7,000 of these data for our research, excluding rankings that did not include specific viewpoints.

C. Experimental Tasks

A subject experiment was conducted online in which participants were asked to label the relevance between the query and the movie. Ten participants performed the labeling. First, participants were given a link to a premade Google spreadsheet. Each sheet contained a query with up to 40 movie titles, one for each row. This is a mixture of the top 10 movie rankings output by each of the four methods for a single query. The participants rated how closely each movie related to the query on a 5-point scale from one to five. Participants were allowed to search for movie information on the Internet if they were not familiar with the movie.

We prepared 10 queries in advance. The actual queries are shown in Table II. Two participants labeled 80 movies for two queries.

D. Implementation

We implemented a system that could calculate the relevance between a given query and movies for each of the comparison methods. The system inputs a query and output the top 10 rankings of each method. The BERT Japanese pretrained model[2] was used as the BERT model to vectorize the query. MeCab[3], a famous Japanese morphological analyzer was used as the tokenizer for the BERT vectorization. As a vocabulary dictionary for MeCab, mecab-ipadic-neologd[4], which supports new words and proper expressions, was used. Keras, a Python library for neural networks, was used for implementing the ranking models.

[2]Kurohashi Lab. Kyoto University: https://nlp.ist.i.kyoto-u.ac.jp/EN/

[3]MeCab: Yet Another Part-of-Speech and Morphological Analyzer: https://taku910.github.io/mecab/

[4]mecab-ipadic-NEologd: Neologism dictionary for MeCab https://github.com/neologd/mecab-ipadic-neologd

TABLE III
PRECISION AT k AND NDCG FOR EACH METHOD

	p@1	p@5	p@10	nDCG
Proposed Method	0.50	0.56	0.58	0.64
Metadata Only	0.50	0.46	0.40	0.60
Movie Similarity	0.20	0.24	0.27	0.47
Review Sentence Similarity	**0.80**	**0.74**	**0.72**	**0.75**

E. Experimental Results

This section describes the results of the subject experiment. We compared the methods by using three metrics: P@k for the precision of each method, nDCG (normalized Discounted Cumulative Gain) for the ranking accuracy, and the participants' average rating.

First, the top part of the ranking of each method was evaluated according to precision. Precision is a measure of how correct the answers were included at the top of the ranking. In this experiment, a rating of three or higher on the participant's 5-point scale was considered correct, and the accuracy was calculated based on this number of correct answers. p@k (precision at k) represents the precision of the movies up to the k position in the ranking.

The precision for each method is shown in Table III. Review Sentence Similarity was the most accurate p@1, p@5, and p@10. The proposed method was ranked second for all p@1, p@5, and p@10. Movie Similarity had the lowest accuracy for all p@1, p@5, and p@10.

Next, nDCG was used to check the accuracy of the ranking. The nDCG for each method is shown in Table III. The most accurate method was the Review Sentence Similarity, with a score of 0.75. In contrast, the Movie Similarity method had the lowest nDCG score of 0.47.

F. Ratings by Participants for Each Query

In this experiment, participants were asked to rate on a 5-point scale how closely a movie related to a given query. The average of the ratings by the participants for each method and each query is shown in Table IV. For the queries "Tearjerker," "Suspense," "Suitable for dating," and "Laughable," the proposed method was the most accurate in finding the highest-rated movies. For the queries "Anime," "Shocking," and "Takeshi Kitano," the method using Metadata Only was the most accurate. For all other queries, the Review Sentence Similarity method was the most accurate.

As an example of the output rankings, Table V shows the top 10 movies for the query "tearjerker" by the proposed and the baseline methods. In this result, participants labeled all movies found by the proposed method as "tearjerker" movies (*i.e.,* all movies were rated at least three). The proposed method seemed to be well-suited for searches based on the viewer's impression of the movie as a whole.

In contrast, as an example of the case of the proposed method not working well, Table VI shows the ranking for the

TABLE IV
AVERAGE PARTICIPANTS' RATING OF THE MOVIES IN THE TOP 10 RESULTS FOR QUERIES (5-POINT SCALE, 1 TO 5)

	Tearjerker	Animation	Suspense	Ghibli	Suitable for dating	Suitable for children	Shocking	Surprise ending	Takeshi Kitano	Laughable
Proposed Method	**3.7**	2.2	**3.6**	1.6	**3.5**	3.8	2.4	3.9	1.5	**3.0**
Metadata Only	2.1	**3.8**	3.1	1.9	1.9	1.7	**3.7**	2.9	**4.0**	1.7
Movie Similarity	2.1	2.2	1.7	1.7	2.9	2.7	2.4	2.9	1.8	1.7
Review Sentence Similarity	2.9	2.6	**3.6**	**4.3**	**3.5**	**4.2**	3.4	**4.5**	3.6	1.7

TABLE V
EXAMPLE OF RESULTS THAT THE PROPOSED METHOD WORKED WELL (FOR THE QUERY "TEARJERKER," RATING BY PARTICIPANTS WAS ON A 5-POINT SCALE, 1 TO 5.)

Proposed Method		Movie Similarity	
Movie title	Rating	Movie title	Rating
What Dreams May Come	3.0	PPiL GU	2.0
Gray Sunset	3.5	Yellow Hair	1.5
Glory Daze	3.0	Columbo Goes to the Guillotine	2.0
The Boy Who Could Fly	4.0	Lightereul kyeora	1.5
Crayon Shin-chan: The Adult Empire Strikes Back	4.5	Nan va Koutcheh	2.0
Pay It Forward	3.5	New Deka Matsuri	1.5
It's a Wonderful Life	4.0	Human Traffic	3.0
Jack	3.5	JUNGLE JUICE	2.0
Life is Beautiful	4.0	Rainy Dog	3.0
The Notebook	4.0	Bad Reputation in the Marketplace	2.0

query "Takeshi Kitano" (note that because both the query and dataset were in Japanese, many Japanese movies were included in the search results). In this case, the Metadata Only method was highly accurate, but methods using review did not work well. The proposed method found many animation movies not related to the query.

V. DISCUSSION

This section discusses the experimental results, the effectiveness of the methods, and which methods are suitable for each query and task. Overall, the proposed method was more accurate for retrieval than the cosine similarity or metadata methods. However, a simple similarity calculation between a single sentence in a movie review and a keyword query was even more accurate.

One possible cause of this was the limitation of representing the entire movie by a single vector. If all reviews for a movie were vectorized and summarized by pooling, the overall trend of the movie (*i.e.,* sad, funny, and warm) might remain as a feature. However, individual scenes and minor impressions would be diluted when summarizing many opinions. Even under these conditions, the proposed method was more accurate than simple cosine similarity. The vector properties were different between a vector of short keyword queries and a vector of longer reviews from multiple people. Therefore, the method of calculating the validity of the combination using a neural network would have been effective, since simply taking the cosine similarity would not correctly calculate the similarity.

The method of Review Sentence Similarity is considered to be more accurate because a highly granular variance representation can be obtained from the review sentences. The nDCG had the highest Review Sentence Similarity, confirming the effectiveness of using reviews. This was probably because, in our experimental search task, it was more important whether there was an element applicable to the query than the overall trend of the movie. For example, whether the movie has a surprise ending depends not on the movie as a whole, but on a single scene. In this case, it was possible to determine whether the review included a description of the turnover scene without reading reviews for the entire movie. In such a case, it would simply be more accurate not to characterize the entire movie.

The scores for each query indicated that the effective methods varied depending on the query type. The metadata-based methods scored higher for queries for which there was likely to be information in the metadata, such as "Animation," "Shocking," and "Takeshi Kitano." However, the review-based method scored higher for queries that describe the nature of the movie, such as "Tearjerker," "Laughable," "Suitable for dating," and "Surprise ending." This shows that using reviews makes it possible to search for movies based on information that does not exist in the metadata.

VI. CONCLUSION

This paper proposes a method for calculating the relevance between a certain keyword query and movies based on user-generated rankings and movie reviews. The model learned the relationship between words in the ranking title and movie reviews using neural networks that used data taken from a user-generated movie ranking forum. Learning through the task of estimating whether a movie is included in the ranking with a given title enabled the model to rank movies.

To verify the usefulness of the model, we conducted subject experiments. Participants were asked to score the relevance between the given queries and movies ranked high by each comparison method. Experimental results showed that the proposed method, which uses user-generated reviews, was more accurate than traditional simple cosine similarity; however, it was even more accurate to split reviews into sentences and

TABLE VI
EXAMPLE OF A CASE WHERE THE METADATA ONLY METHOD WORKED WELL (FOR THE QUERY "TAKESHI KITANO," RATING BY PARTICIPANTS WAS ON A 5-POINT SCALE, 1 TO 5.)

Proposed Method		Metadata Only	
Movie title	Rating	Movie title	Rating
Crayon Shin-chan: Unkokusai's Ambition	1.0	Kikujiro	4.5
Case Closed: Captured in Her Eyes	1.0	Sonatine	5.0
Godzilla, Mothra and King Ghidorah	1.5	Boiling Point	3.5
Air	2.0	Zatoichi	4.0
Lupin III: Hemingway Papers	1.5	Beyond Outrage	4.0
Iki-jigoku	2.0	Outrage Coda	4.0
Mosquito	1.0	BROTHER	3.5
Star Wars: Episode V	1.0	TAKESHIS'	4.5
RahXephon	2.0	HANA-BI	3.0
Lupin III: The Castle of Cagliostro	1.5	OUTRAGE	4.0

use the maximum similarity method for each of them. We also found that for some tasks, using metadata was more accurate.

We plan to use the proposed model to determine whether a given movie appears in a given titled ranking or not. Specifically, we are considering using this as a fine-tuning task for the language model.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grants Numbers 21H03775, 21H03774, and 22H03905.

REFERENCES

[1] T. Miyashita, Y. Shoji, S. Fujita, and M. J. Dürst, "Movie keyword search using large-scale language model with user-generated rankings and reviews," in *The 25th International Conference on Information Integration and Web Intelligence*, 2023, pp. 249–255.

[2] J. Ramanand, K. Bhavsar, and N. Pedanekar, "Wishful thinking - finding suggestions and 'buy' wishes from product reviews," in *Proc. of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 2010, pp. 54–61.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[5] Z. Liu, D. Huang, K. Huang, Z. Li, and J. Zhao, "Finbert: A pre-trained financial language representation model for financial text mining," in *Proc. of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020, pp. 4513–4519.

[6] W. Sakata, T. Shibata, R. Tanaka, and S. Kurohashi, "Faq retrieval using query-question similarity and bert-based query-answer relevance," in *Proc. of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 1113–1116.

[7] W. Yang, H. Zhang, and J. Lin, "Simple applications of bert for ad hoc document retrieval," *arXiv preprint arXiv:1903.10972*, 2019.

[8] Y. Shao, J. Mao, Y. Liu, W. Ma, K. Satoh, M. Zhang, and S. Ma, "Bert-pli: Modeling paragraph-level interactions for legal case retrieval." in *IJCAI*, 2020, pp. 3501–3507.

[9] Z. Jiang, A. El-Jaroudi, W. Hartmann, D. Karakos, and L. Zhao, "Cross-lingual information retrieval with BERT," in *Proc. of the workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, 2020, pp. 26–31.

[10] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, pp. 225–331, 2009.

[11] A. Soleimani, C. Monz, and M. Worring, "Bert for evidence retrieval and claim verification," in *Advances in Information Retrieval*, J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, Eds., 2020, pp. 359–366.

[12] L. Yu, K. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," *Proc. of the Deep Learning and Representation Learning Workshop: NIPS-2014*, 2014.

[13] F. F. Faria, A. Veloso, H. M. Almeida, E. Valle, R. d. S. Torres, M. A. Gonçalves, and W. Meira, "Learning to rank for content-based image retrieval," in *Proc. of the International Conference on Multimedia Information Retrieval*, 2010, pp. 285–294.

[14] S. K. Karmaker Santu, P. Sondhi, and C. Zhai, "On application of learning to rank for e-commerce search," in *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 475–484.

[15] B. Shaw, J. Shea, S. Sinha, and A. Hogue, "Learning to rank for spatiotemporal search," in *Proc. of the Sixth ACM International Conference on Web Search and Data Mining*, 2013, pp. 717–726.

[16] K. Kurihara, Y. Shoji, S. Fujita, and M. J. Dürst, "Learning to rank-based approach for movie search by keyword query and example query," in *The 23rd International Conference on Information Integration and Web Intelligence*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 137–145.