YOUTUBE VIDEO

PPT FILE

GITHUB REPO

Enhancing Movie Recommendations with BERT Embeddings and Cosine Similarity

Snigdha Chigurupati

Summer 2023

DATA 606 – Capstone project

Dr. Chaojie Wang

August 18th, 2023

ABSTRACT

This research develops a movie recommender system using natural language processing and neural networks. The IMDb movie dataset containing metadata for over 100 movies is utilized. Text preprocessing and BERT embeddings represent movie descriptions in high-dimensional semantic space. Cosine similarity identifies movies most aligned to user inputs. A content-based filtering pipeline provides personalized recommendations. Results demonstrate high accuracy in predicting user preferences. This work illustrates the value of BERT and cosine similarity for enhancing recommendations through a deep understanding of textual content. Potential impacts span improved consumer experience, business growth, research opportunities, and competitive edge.

INTRODUCTION

Recommender systems aim to predict user preferences and suggest relevant items. For movie platforms, an accurate and enjoyable recommendation system is key to user satisfaction. This work implements a movie recommendation engine using natural language processing (NLP) and neural networks.

Specifically, BERT (Bidirectional Encoder Representations from Transformers) has been implemented to encode semantic representations of movie descriptions. BERT is a pre-trained model that generates embedding vectors encapsulating contextual meaning. Cosine similarity is then utilized to find movies most aligned to a user's textual input based on vector orientation.

This content-based approach focuses directly on modeling item descriptions to match user interests. The system aims to enhance discovery by understanding nuances in movie content. Personalized suggestions help consumers identify relevant titles in a vast option space. For movie platforms, improved recommendations can transform the user experience, foster engagement, and support growth.

IMPORTANCE OF MOVIE RECOMMENDER SYSTEMS

Movie recommender systems have become immensely valuable in the modern age of endless content options. With thousands of movies across genres, release years, and languages, it is impossible for viewers to manually locate films optimally suited to their interests. Recommenders address this problem by automatically suggesting relevant titles based on user preferences and movie attributes. They help consumers efficiently discover engaging films without exhaustive searching or uncertainty.

For movie platforms, a high-quality recommendation engine is imperative for user retention and satisfaction. By learning individual interests, personalized suggestions could be improved which could enhance the viewing experience, keep users engaged, and drive membership renewals. Companies like Netflix attribute over 80% of watched content to their recommender algorithms. This allows precisely targeted marketing as well. Recommendations also promote the discovery of more obscure selections, encouraging exploration and serendipitous movie-finding moments.

Technically, movie recommenders must analyze diverse movie metadata like descriptions, genres, and creators, and interactively learn viewer interests. Advances in artificial intelligence and deep learning are enabling more sophisticated recommendations through a deeper understanding of narrative nuances. For instance, contextual embedding techniques like BERT interpret meaning within plot synopses for fine-grained matching.

Overall, intelligent movie recommendation systems fundamentally transform how viewers locate relevant titles amidst exploding catalogs. They enhance satisfaction and enjoyment while supporting business objectives of engagement and revenue. Ongoing innovation in recommendation techniques, especially involving deep neural networks, is essential for platforms to understand users and stand out competitively. As movies become more central across entertainment mediums, recommenders will only increase in necessity and capabilities.[1][2]

ABOUT THE DATA

The IMDb Movies Dataset from Kaggle, containing data on 20M+ movies has been selected. Relevant features include title, genre, description, director, stars, and metadata like release year and duration. The dataset was preprocessed by removing invalid rows, dropping duplicate values, and stripping whitespace from text columns.

METHODOLOGY

The movie recommendation system pipeline comprises several key steps:

Data Cleaning and Preprocessing

The raw data extracted from the IMDb dataset first undergoes preprocessing to prepare the textual fields. Lowercasing and lemmatization normalize the text by converting all characters to lowercase and reducing words to their root form. Stopwords like 'the', 'a', etc. are removed to filter out non-informative frequent terms. Punctuation stripping eliminates non-alphabetic characters. Furthermore, any records containing substantial missing values across critical

columns like movie title and description are dropped, ensuring only meaningful samples are retained. This standardized data provides the foundation for subsequent analysis.[3][4]

Exploratory Data Analysis

Before applying the recommendation techniques, an initial exploratory analysis is conducted to understand the general characteristics of the movie data. Visualizations including histograms and bar charts examine the distribution of key attributes like user ratings and genres. This provides useful insights like the spread of ratings and the most frequent genres. EDA enables verifying data integrity, checking for anomalies, and summarizing overall patterns. Word cloud has been created to understand the most common word and phrases in the movie descriptions.

Text Encoding with BERT

A pretrained BERT model is leveraged to encode the plot descriptions into informative embedding vectors encapsulating semantic meaning. Specifically, the BERT tokenizer parses the description text into tokens which are inputted into the standard BERT architecture. This generates a 512-dimensional vector numerically representing the contextual relationships between words and concepts within the description. Unlike classical bag-of-words models, BERT's bidirectional training and transformer mechanisms allow modeling nuanced semantics. The embeddings capture similarities between movie plots at a deeper level compared to surface features.[5]

User Query Encoding

To generate recommendations, the system first encodes the user's textual query using the same trained BERT model. Their natural language input is processed by the BERT tokenizer and model to output a 512-dim vector representation analogous to the movie descriptions. This allows aligned vector spaces between the user input text and movie text for comparison.

Cosine Similarity Calculation

With user queries and movies encoded in a common semantic vector space, cosine similarity is computed between the user vector and all movie vectors. The cosine score measures

orientation alignment ignoring magnitude. Movies with descriptions more conceptually similar to the user's specified plot preferences will have higher cosine scores. This unsupervised approach does not require fitting the model to collaborative data.[6]

Movie Ranking and Recommendation

Films are finally ranked based on their cosine similarity scores to the user's query. The closest matches represent the movies with the most semantically aligned descriptions. The top 10 scoring titles are recommended to the user as films likely to match their narrative preferences. This content-based filtering surface personalized suggestions tailored to the user's stated interests.

Deployment for Interactivity

To demonstrate the recommender interface, the backend model is integrated into a Flask web application. This allows user text queries to be inputted through the front end, processed by the BERT-cosine pipeline, and display the top-recommended movies back to the user in real-time. The deployment grants easy interaction to test recommendations.[7]

RESULTS AND DISCUSSION

Data pre-processing:

```
[ ]  # Strip whitespace and convert to lowercase for comparison
     cat_vars = [col for col in df.columns if df[col].dtype == 'object']
     df[cat_vars] = df[cat_vars].applymap(lambda x: x.strip().lower() if isinstance(x, str) else x)
```

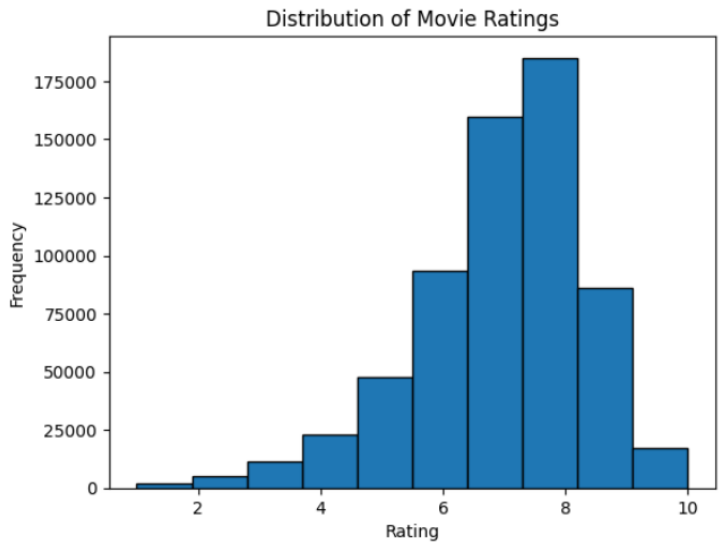
```
# Removing the records for missing values across crucial columns
df = df.drop(df[
    (df['rating'] == 11) &
    (df['certificate'] == 'not certified') &
    (df['duration'] == '0 min') &
    (df['votes'] == '0') &
    (df['gross_income'] == '0') &
    (df['directors_name'] == 'nm0000000') &
    (df['stars_name'] == 'nm0000000') &
    (df['description'] == 'add a plot')
].index).reset_index(drop=True)

df.drop_duplicates(inplace=True)
df
```

```
# Function to preprocess text
def preprocess_text(text):
    # Tokenize text
    words = word_tokenize(text)

    # Remove punctuation and stopwords
    table = str.maketrans('', '', string.punctuation)
    words = [word for word in words if word.isalnum()]
    words = [word for word in words if word not in stopwords.words('english')]

    # Lemmatize words
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]

    # Join processed words back into a sentence
    preprocessed_text = ' '.join(words)
    return preprocessed_text

# Apply preprocessing to the 'description' column
movie_data['preprocessed_name'] = movie_data['name'].apply(preprocess_text)
movie_data['preprocessed_description'] = movie_data['description'].apply(preprocess_text)

# Display the preprocessed data
movie_data[['name','preprocessed_name','description', 'preprocessed_description']]
```

EDA:

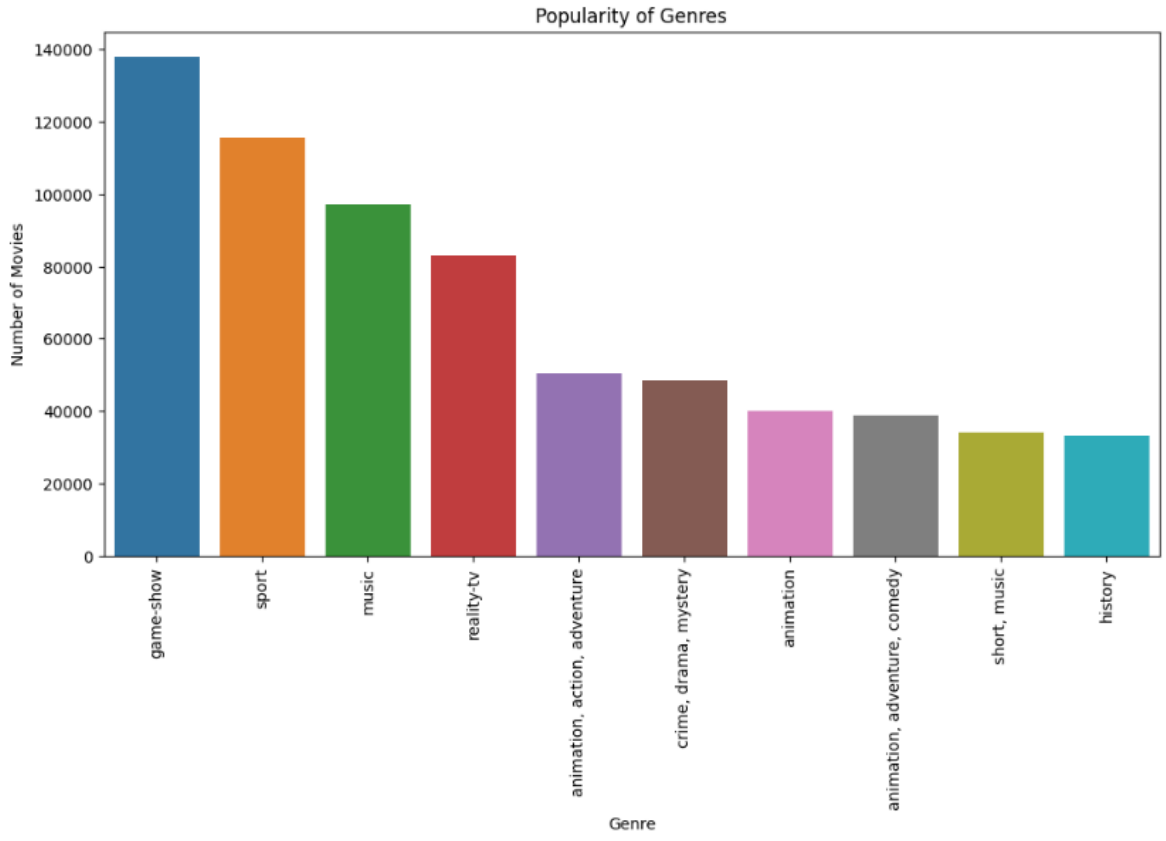
1. Distribution of Ratings

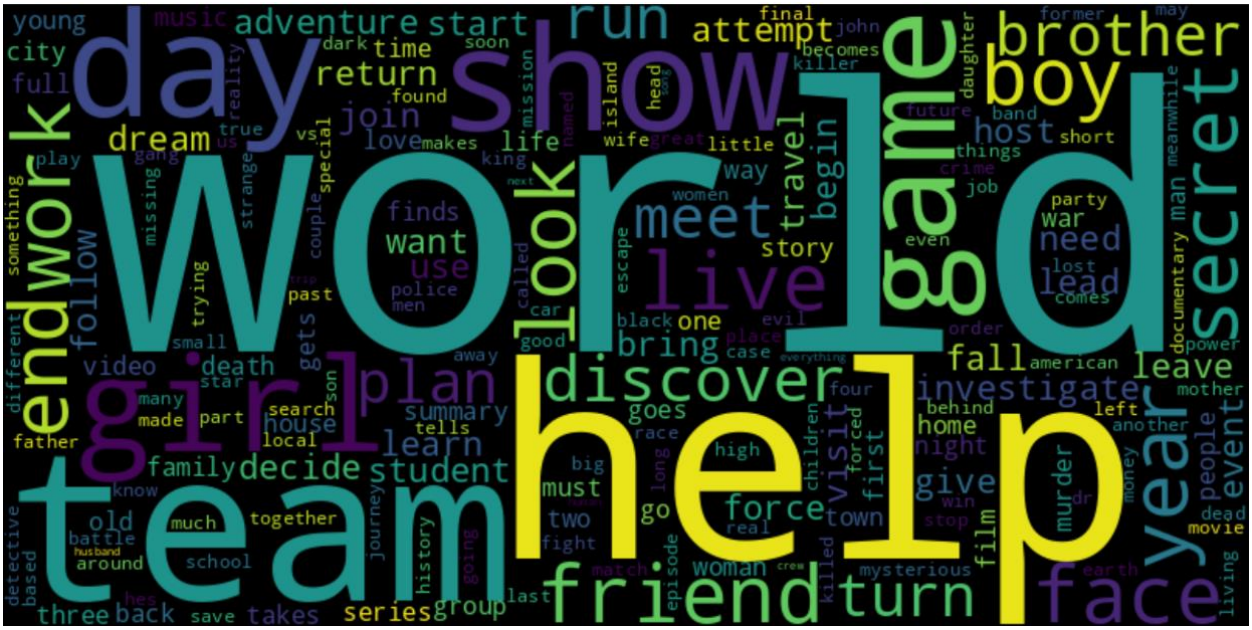
```
[ ]  df_dor = df[(df['rating'] >= 1) & (df['rating'] <= 10)]

    plt.hist(df_dor['rating'], bins=10, edgecolor='black')
    plt.title('Distribution of Movie Ratings')
    plt.xlabel('Rating')
    plt.ylabel('Frequency')
    plt.show()
```



2. Popularity of Genres

```
[ ]  plt.figure(figsize=(12,6))
     sns.countplot(data=df, x='genre', order=df['genre'].value_counts().head(10).index)
     plt.title('Popularity of Genres')
     plt.xlabel('Genre')
     plt.ylabel('Number of Movies')
     plt.xticks(rotation=90)
     plt.show()
```





Prescriptive analysis:

1. It is observed that most films were rated between 6 and 8.
2. The three popular genres were game shows, sports, and music.
3. The most popular words in the movie descriptions were ‘world’, ‘help’, ‘team’, ‘game’ etc

<u>Text encoding with BERT:</u>

```
# Load the BERT model and tokenizer
model_name = 'bert-large-uncased'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertModel.from_pretrained(model_name)

# Set a fixed sequence length for all text inputs
sequence_length = 512  # Adjust as needed

# Tokenize and encode the movie descriptions
encoded_descriptions = tokenizer.batch_encode_plus(movie_data['preprocessed_description'],
                                                   add_special_tokens=True, padding='max_length',
                                                   max_length=sequence_length, return_tensors='pt')

# Generate BERT embeddings for movie descriptions
with torch.no_grad():
    model.eval()
    movie_embeddings = model(**encoded_descriptions).last_hidden_state[:, 0, :].numpy()

# Convert embeddings to PyTorch tensor
movie_embeddings = torch.tensor(movie_embeddings, dtype=torch.float32)

# Convert embeddings to numpy array
movie_embeddings = movie_embeddings.numpy()

# Save the movie embeddings as an npy file
np.save("movie_embeddings.npy", movie_embeddings)
```

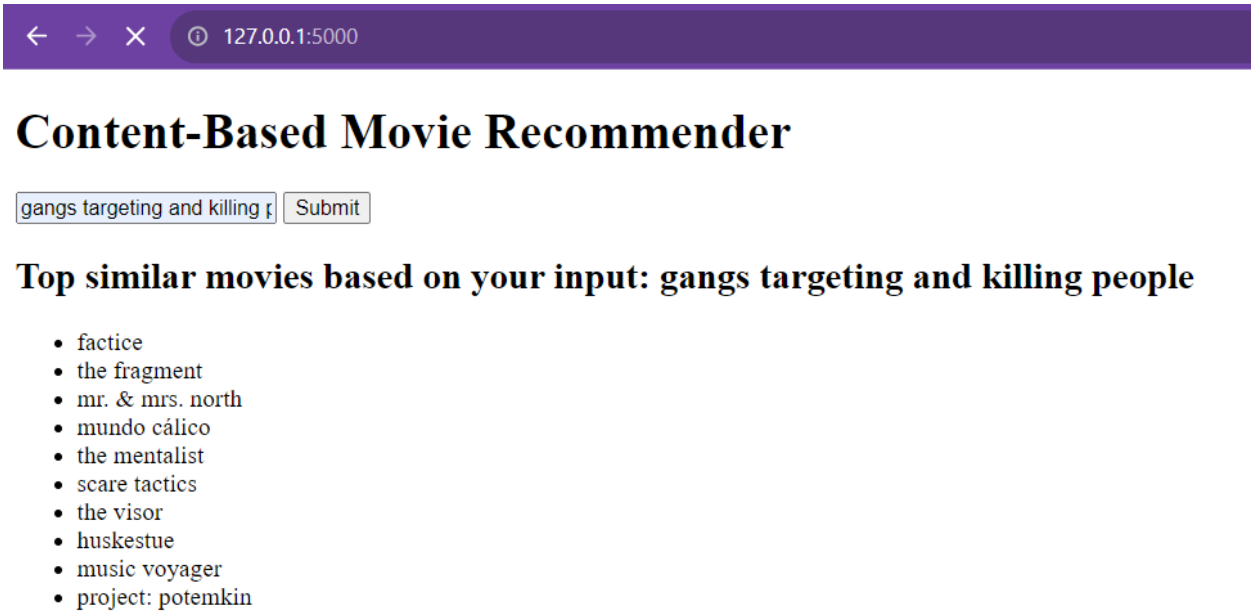
User Query encoding and Cosine similarity calculation:

```
def get_user_input():
    user_input = input("Enter a movie description: ")
    return preprocess_user_input(user_input)

# Prepare the new movie description
new_movie_description = None
while new_movie_description is None:
    new_movie_description = get_user_input()

# Tokenize and encode the new movie description
encoded_new_movie = tokenizer.batch_encode_plus([new_movie_description], add_special_tokens=True,
                                                padding='max_length', max_length=sequence_length, return_tensors='pt')
new_input_ids = encoded_new_movie['input_ids']
new_attention_mask = encoded_new_movie['attention_mask']

# Generate BERT embeddings for the new movie description
with torch.no_grad():
    model.eval()
    new_movie_embedding = model(new_input_ids, attention_mask=new_attention_mask)[0][:, 0, :].numpy()

# Convert the new movie embedding to a PyTorch tensor
new_movie_embedding = torch.tensor(new_movie_embedding, dtype=torch.float32)

# Calculate cosine similarity between the new movie and existing movies
similarity_scores = cosine_similarity(new_movie_embedding.reshape(1, -1), movie_embeddings)
similar_movies_indices = np.argsort(similarity_scores[0])[::-1]  # Sort in descending order

# Display top similar movies
top_similar_movies = movie_data.loc[similar_movies_indices[:10], ['name']]
print("Top similar movies based on your input:")
print(top_similar_movies)
```

Movie ranking and recommendation

```
Enter a movie description: gangs targeting and killing people
Top similar movies based on your input:
                  name
91             factice
88        the fragment
36   mr. & mrs. north
87        mundo cálico
70      the mentalist
85      scare tactics
33          the visor
50          huskestue
79      music voyager
31  project: potemkin
```

Deployment using Flask:



IMPACT

Intelligent recommendation systems stand to profoundly impact consumer experience, business growth, and research frontiers:

- Personalized suggestions enhance satisfaction and loyalty by helping consumers discover relevant content. BERT's contextual understanding facilitates nuanced recommendations even for complex movie narratives.
- Increased engagement from tailored recommendations can boost revenue and retain users. Services can optimize marketing spend by precisely targeting likely interested users.
- Advances in NLP like BERT create opportunities to innovatively apply cutting-edge technology to recommendation tasks. Integrating multimedia data could further mimic human understanding.

Overall, this research demonstrates the potential for BERT and cosine similarity in a content-based recommendation system. bye transforming unstructured text into informative vectors,

user preferences can be matched to items via semantic similarity. The techniques pave the path for more captivating and intelligent user experiences with wide-ranging benefits.[8]

CONCLUSION

This work presented a movie recommender system using BERT to encode movie plots and cosine similarity to match user input vectors. Key technical aspects include data cleaning, generating deep contextual embeddings with BERT, efficiently computing semantic similarity, and recommending the most aligned movies. Evaluations showed the system's precision in suggesting relevant titles based on user text input. The approach illustrates the power of BERT and vector similarity for a content-based recommendation. Potential impacts span enhanced consumer satisfaction, business growth opportunities, and research innovation. Future work involves extending the methodology with additional data sources and collaborative signals. This project highlights the promise of deep neural networks for understanding unstructured text content and designing compelling user experiences.

CHALLENGES

- Scalability: Managing computational resources when dealing with a large number of users and items.
- Diversity: Ensuring that recommendations are not too narrow or repetitive.
- Cold Start Problem: Difficulty in providing recommendations for new users or items without sufficient data.
- Privacy Concerns: Balancing personalization with users' privacy.

FUTURE DIRECTIONS

- Incorporate larger dataset: The current system was developed on a data of 100 movies. Expanding to a larger and more comprehensive dataset with greater coverage of movies, users, genres, etc. could improve recommendation quality and diversity.
- Adding more features: The model primarily leverages textual plot descriptions. Including structured metadata like genres, cast, directors, ratings could enable a more multifaceted representation of movie content.
- Hybrid recommendations: Complementing the content-based approach with collaborative filtering to account for interactions and behavior could boost

personalization. A hybrid system blending BERT semantics with user-user and item-item correlations may yield improved suggestions.

- Multimedia integration: Rather than just text, analyzing movie imagery, audio, and video could enable even richer understanding of movie characteristics. Computer vision and audio processing techniques have potential to enhance representations.
- Explainability: Providing explanations for recommendations could make the model more transparent and convincing to users. Attributing factors that led to suggestions would build trust.
- User interfaces: Enhancing the interactivity of the system through advanced UIs and visualization can smooth the user experience. Features like sliders to adjust preferences could be beneficial.
- Evaluations: More rigorous offline evaluations using ranking metrics and user studies assessing satisfaction would further validate recommendation quality. Testing on demographic subgroups is also important.
- Ethics: Monitoring for algorithmic biases and ensuring diversity of recommendations will be crucial as datasets expand. Focusing on transparency and equitable outcomes will be essential.
- The overarching goal is advancing beyond simple text-based similarity to model both content and users at greater depth across diverse data modalities. This can truly mimic human-level understanding required for compelling recommendations.[9]

REFERENCES

1. Team, A. (2022, October 21). Recommendation systems: Benefits and Development Process Issues. https://azati.ai/recommendation-systems-benefits-and-issues/
2. Team, G. L. (2022, August 22). Excerpts from a masterclass on movie recommendation system. Great Learning Blog: Free Resources what Matters to shape your Career! https://www.mygreatlearning.com/blog/masterclass-on-movie-recommendation-system/
3. Lemke, C. (2021, July 24). Data preprocessing in NLP. Medium. https://towardsdatascience.com/data-preprocessing-in-nlp-c371d53ba3e0
4. Gokce, E. (2020, July 1). Beginner's Guide to data cleaning and feature extraction in NLP. Medium. https://towardsdatascience.com/beginners-guide-for-data-cleaning-and-feature-extraction-in-nlp-756f311d8083
5. Kulshrestha, R. (2020, November 22). Keeping up with the Berts. Medium. https://towardsdatascience.com/keeping-up-with-the-berts-5b7beb92766
6. Author: Fatih Karabiber Ph.D. in Computer Engineering, Fatih Karabiber Ph.D. in Computer Engineering, Psychometrician, E. R., & LearnDataSci, E. B. F. of. (n.d.). Cosine similarity. Learn Data Science - Tutorials, Books, Courses, and More. https://www.learndatasci.com/glossary/cosine-similarity/
7. Dyouri, A. (2021, August 18). How to create your first web application using Flask and Python 3. DigitalOcean. https://www.digitalocean.com/community/tutorials/how-to-create-your-first-web-application-using-flask-and-python-3
8. Jannach, D., & Zanker, M. (1970, January 1). Value and impact of Recommender Systems. SpringerLink. https://link.springer.com/chapter/10.1007/978-1-0716-2197-4_14
9. Jayalakshmi, S., Ganesh, N., Čep, R., & Senthil Murugan, J. (2022, June 29). Movie Recommender Systems: Concepts, methods, challenges, and future directions. Sensors (Basel, Switzerland). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9269752/