**MOVIE RECOMMENDATION SYSTEM USING KNOWLEDGE GRAPH AND CONTENT BASED FILTERING WITH SENTIMENT ANALYSIS**

ABSTRACT

The recommendation system based on sentiment analysis is proposed for the goal of helping users to decide on movies. The aim of this work is to find both recommendation system techniques and sentiment analysis in order to generate the most accurate recommendations for users. Because both domains suffer from the lack of labelled data. Hence in our analysis we provide findings on the sentiment analysis with text data of Metacritic movie dataset and to find the positive and negative reviews using NLP techniques. Since content-based filtering uses features and its previous actions to recommend users, we use sentiment analysis and other features for recommendation. Knowledge graph is implemented using neo4j database and GraphDB with different querying techniques SPARQL.

1. **Introduction**

Knowledge graphs are a powerful technology which is useful in various domains. The queries can retrieve the information and the information explicable to the users. Natural language processing, machine learning techniques are used in the movie review datasets to analyse the sentiment and other factors. Section 1 is an introduction to the project. Section 2 describes the implementation of the project. Section 3 presents the detailed process of implementing the knowledge graph. Section 4 deals with evaluation. Section 5 consists of Conclusion and future work.

1. **Implementation**
   1. **Importing dataset**

The Metacritic dataset is imported from the metacritic movie website and uploaded to the Google Colab notebook. In this project, we use the Colab platform for implementation.

**2.2 Data Pre-Processing**

The imported data is pre-processed with data cleansing techniques to check null values, unstructured data and to form the structured data.

* 1. **NLP For Sentiment Analysis**

Natural Language Processing is performed for the pre-processed movie review text data to find the sentiment analysis. First replace the punctuations with space. Replace or remove the short words. Making entire text as lowercase brings more convenient. Remove the stop words by importing stopwords and word tokenizer from nltk package. Lemmatization is a text normalization which converts the word into meaningful base form. Here we import Wordnet and WordNetLemmatizer from nltk package. Most frequent words can be viewed by bar graph. Word Cloud is used to visualize the data in which the size of each word represents its frequency or importance.

The processed text is reviewed to find sentiment analysis. Vadersentiment is the pre-defined python package with SentimentIntensityAnalyser which identifies the sentiments. Using this, we get the total count of positive, negative and neutral reviews. This helps the users to get the recommendation according to their sentiments. Importing pipeline from transformer extracts the emotions from the text data. This brings the most accurate emotion of the user to suggest the same.

* 1. **Content-Based Filtering**

Content-based filtering is based on TF-IDF matrix and the cosine similarity score to evaluate the similarities between the features. Term Frequency-Inverse Document Frequency is a text vectorizer that transforms the text into a usable vector. The term frequency is the number of occurrences of a specific term in a document. Cosine similarity measures the similarity between the two vectors. It calculates the cosine of the angle between two vectors projected in a multi-dimensional space.

In this project, we use content-based filtering for each feature, which brings the recommendation based on user’s requirement. Using sci-kit learn package, we import feature-extraction with TfidVectorizer to perform TF-IDF and import sigmoid\_kernel from sklearn.metrics to perform cosine similarity. Based on similarity between both matrices the movie is recommended. Different recommendations based on different features like movie description and title, genre, actor, and the most importantly based on sentiment, the movie recommendation has made using these two techniques.

1. **Knowledge Graph**

Knowledge graphs are used to store interlinked descriptions of entities-objects, events and relationships.Knowledge Graph is implemented using neo4j graph database. Neo4j is an open source which has flexible schema data model, real-time insights, high availability with transactional guarantees, easy retrieval with Cypher query language. By importing spacy and matcher we create entities and relationships based on subject, verb and object from the pre-processed text data. Source, target and relations are created as a dataframe.

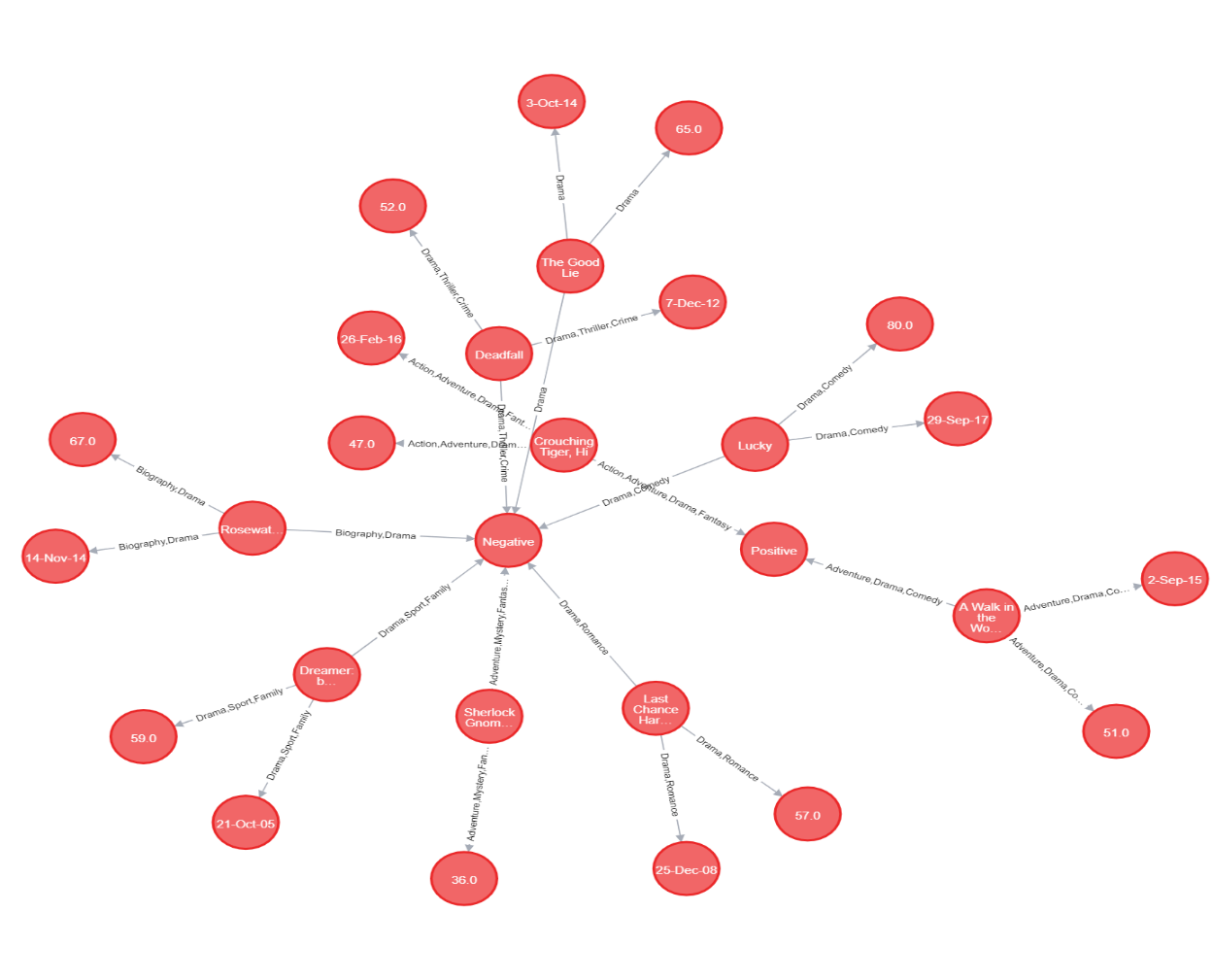
* 1. **Neo4j Connection**

Neo4j Graph Database is installed and import the neo4j tqdm. Using NERS name entity recognition, the entities and relationships are defined in the neo4j connection. The neo4j sandbox is created and the URI, Password, Username given in the website have to be entered in the colab to connect with the neo4j graph database. The below given is the query to link the nodes in the neo4j graph.

MERGE (n1)-[:LINKS{relations:row.source}]->(n2)

* 1. **Neo4j Graph Visualization**

The node entities and relationships are connected with links as a graph.

****

Sample Cypher queries in neo4j graph to return the nodes matching the query

MATCH (n1)-[r]-(n2) WHERE n1.name=” The Hustler” RETURN n1,r,n2

* 1. **Knowledge graph with GraphDB using multiple ontologies**
     1. Knowledge Creation

The existing movie datas from Wikidata as well as from the metacritic were used to access the links and to the DBPedia.

* + 1. Knowledge hosting

GraphDB was used as a TripleStore as it provides a simple way of uploading .ttl files, and interface for testing SPARQL queries.

* + 1. Knowledge assessment

Information about the IMDB and the metacritic was gathered from the website since different movies have different URLs, the movie’s different IDs were to be retrieved. In Wikidata Query service, the following query retrieves the list of all IMDb IDs and Metacritic IDs.

SELECT ?item ?imdb ?metacritic

WHERE

{

?item wdt:P345 ?imdb.

OPTIONAL { ?item wdt:P1712 ?metacritic }

}

DBPedia maintains links in the form of owl:sameAs relations to Wikidata. SPARQL query enables linking to Wikidata ny using the sameAs:

PREFIX schema: < http://schema.org/>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> SELECT ?film ?wdlink ?dblink WHERE {

SERVICE <http://dbpedia.org/sparql> {

?dblink a dbpedia-owl:Film.

?dblink owl:sameAs ?wdlink

}

?film a schema:Movie.

?film owl:sameAs ?wdlink

}

* + 1. Knowledge deployment and visualization

The retrieved data can be saved as RDF file and in the GraphDB, the rdf file (.ttl file) can be uploaded and visualize the knowledge graph. The knowledge graph is stored on a server and the data comprises the extracted movie datas from Metacritic.

1. **Evaluation**

We extracted and analysed 5,83,549 user’s movie reviews and 15,000 movies from metacritic.com and 2,64,897 movies from imdb.com. The sentiments were extracted from VaderSentiment Analysis and nlp packages which trained the datas according to it. The total count of positive, negative and neutral were analysed. In which 4,06,922 reviews were positive, 1,35,999 reviews were negative and 40,629 were neutral. This shows that the maximum reviews of the movies were positive and the people like most of the movies in this website. The emotions were also extracted, particularly, admiration, joy, neutral, amusement, surprise and positive expectation have been present in more numbers. Integrating with a knowledge graph improves both accuracy and interpretability. Using a knowledge graph enables recommender systems to capture the user-item interactions and could be able to provide more accurate recommendations.

1. **Conclusion and future work**

Recommendation system based on a content-based filtering and using knowledge graph has been created. The resulting knowledge graph and the sentiment analysis using feature-extraction provides the better recommendation to the users. Further research can be made in the knowledge graphs in the area of natural language processing, since the emotions detecting for large text datas are time consuming and inaccurate. However, data about the movies was retrieved successfully.

2. Rottentomatoes, available at: https://www.rottentomatoes.com .

**References**

1. A. Breitfuss, K. Errou, A. Kurteva et al, FUTURE 6107 “Representing emotions with knowledge graphs for movie recommendations”, Future Generation Computer Systems (2021)
2. Rottentomatoes, available at: https://www.rottentomatoes.com .

3. Metacritic, available at: https://www.metacritic.com.

4. Twenty Century Fox, Fox Corporation, "20th Century Fox," available at: https://www.foxmovies.com

5. N. S. Koh, N. Hu, E. K. Clemons, "Do online reviews reflect a product's true perceived quality? An investigation of online movie reviews across cultures", In Electronic Commerce Research and Applications, vol. 9, no. 5, pp. 12, Sep 1 2010.

6. IMDb, available at: https://www.imdb.com.

7. Douban, available at: https://www.douban.com.

8. Wikipedia, "Theory of Planned Behaviour", available at: https://en.wikipedia.org/wiki/Theory\_of\_planned\_behavior.

9. Wikipedia, "Theory of Reasoned Action", available at: https://en.wikipedia.org/wiki/Theory\_of\_reasoned\_action.

10. J. Z. Pan, G. Vetere, J.M. Gomez-Perez, H.Wu, "Exploiting linked data and knowledge graphs in large organizations", Heildelberg, Springer, p. 281, 2017.

11. Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734–749, 2005.

12. Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In IJCAI, volume 99, pages 688–693, 1999.