A CONTENT-BASED MOVIE RECOMMENDER SYSTEM BASED ON EXOGENOUS AUGMENTED COMMENT TAG DATA

MASTER THESIS

EMRE ERIM

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

STUDENT NUMBER

20664407

COMMITTEE

dr. Paris Mavromoustakos Blom
dr. Emmanuel Keuleers

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

June 23th, 2023

WORD COUNT

7959

ACKNOWLEDGMENTS

A CONTENT-BASED MOVIE RECOMMENDER SYSTEM BASED ON EXOGENOUS AUGMENTED COMMENT TAG DATA

MASTER THESIS

EMRE ERIM

Abstract

Content-based movie recommendation systems mainly use the information from the movie metadata to provide users with personalized movie recommendations based on their previous selections. The content-based filtering method is one of the two most commonly used recommendation methods. In this paper, we propose using augmented user-extracted comment tags as movie metadata in a content-based recommendation system. The semantics are extracted and converted into a fixed-dimensional vector using Doc2Vec. XGboost is used as the main machine learning model to classify the recommendation of movies to users. Experiments show that the recommendation results are not significantly improved in both accuracy and f1 score. Overall, the study provides a valuable contribution to the field of augmentation techniques for content-based recommendation systems.

1 DATA SOURCE/CODE/ETHICS/TECHNOLOGY STATEMENT

The Movielens 25M data has been acquired from open sources. Work on this thesis did not involve collecting data from human participants or animals. The thesis code can be accessed through the GitHub repository by following the link https://github.com/nickbttm/Movie-CBRS-Thesis-Project.

The adapted code fragments are clearly commented out in the notebook. In terms of writing, the author used Quillbot (https://quillbot.com/grammar-check) to improve spelling and grammar. No other typesetting tools or services were used.

2 INTRODUCTION

2.1 *Context*

Recommender systems (RS) are machine-learning-based systems that assist consumers in discovering new products and services (Ricci et al., 2010). Today, recommender systems are being used in a variety of products and services, including books (Crespo et al., 2011), tourism (Isinkaye et al., 2015), movies (Bobadilla et al., 2010), (Chen et al., 2015), (Diao et al., 2014), (Qu et al., 2013), and music (Yoshii et al., 2008). Lee and Hosanagar (2014) show that the utilization of a recommender system resulted in a 35% increase in sales with respect to the control group (no recommender system). Also, they deduce that recommender systems allow customers greater exploration via cross-selling items and increase the volume of sales for all types of items, including niche ones. In today's world, recommender systems are used in large-scale businesses that are based on information, such as Google (J. Liu et al., 2010), Twitter (Ahmed & Jaidka, 2013), LinkedIn (Kenthapadi et al., 2017), and Netflix (Gomez-Uribe & Hunt, 2015).

Content-Based Recommender Systems (CBRSs) are used to recommend similar items based on the past likes of the user (Lops et al., 2011). CBRS, as a movie recommender system, is used to predict movies for users based on the features of previously liked movies (Reddy et al., 2019).

2.2 *Problem Statement*

CBRSs are used for movie recommendations, which use the features of the movies based on the movie metadata and user-extracted information, such as (Reddy et al., 2019), which use content-based filtering with genre correlation. Also, (Li et al., 2016) have suggested a movie recommender system that uses user-extracted features collected from microblogs and social networks together with default data about movies.

In the relevant research, default characteristics of movies are used as features for RS. However, in this study, the default characteristics of movies are avoided. Instead, we attempt to achieve similar results by solely using the augmentation of comment tag data. To add to what "augmented comment tags in context" means, the augmentation process is defined as finding relevant features about comment tags in selected knowledge bases and adding those features to the items' features. In this way, we explore whether it is possible to extract meaningful or accurate recommendations based only on user-related data. Comment tags will be augmented using the knowledge bases WordNet (Miller, 1995) and Wikidata (Vrandečić &

Krötzsch, 2014). These approaches have been extensively addressed in the methodology and experimental setup sections. Based on the approaches determined for this study, the research question is:

> *To what extent can a content-based movie recommender system based only on user-generated features produce accurate movie recommendations?*

The sub-questions can be listed separately, as follows:

RQ1 *To what extent do the added exogenous features to comment tag data improve the accuracy with respect to a non-augmented baseline?*

RQ2 *Which knowledge base (wordnet or wikidata) performs better in terms of giving better augmentation of comment tags to a model predicting known ratings in the database?*

RQ3 *How is the performance of the candidate models on predicting known ratings for each augmentation alternative?*

2.3 *Motivation*

Previously, De Gemmis et al. (2008) have applied machine learning techniques to the semantics of the features of the items to infer better results for users. Kartheek and Sajeev (2021) tackle cold start and sparcity problems by building a semantic-based hybrid recommender system and using knowledge bases to extract semantics. Similar to this research, we investigate how to build a CBRS by not using item descriptions; instead, we try to extract more information from user-generated features that belong to items. To our knowledge, this approach has not been applied before in this context. Therefore, the work presented in this thesis can be considered novel.

The purpose of this research is to tackle the idea that if it is possible to use only user feedback models, they can be generalized as they contain the necessary information by augmenting user-generated tags if the created models' accuracy is comparable to the baseline. This might potentially be applied to other types of recommender systems for music, books, or educational content, where the item description cannot be retrieved as much as for movies but potentially has a lot of user-generated features. In today's world, the available content is getting extremely large, and it might be hard to find enough information about the content to build a recommender system, such as a video on Youtube or a Reddit post that might be useful for a user. These types of contents offer a lot of comments because of the nature of these social networks, and this study tries to

extract this information to enhance the user experience by matching those contents with other potential users.

2.4 *Societal Relevance*

CBRSs derive a list of recommended items for a user based on their previous interactions. The aim of traditional CBRS algorithms is to match the characteristics of the user profile with the features of the recommended items based on the user's interests (Pazzani & Billsus, 2007).

The content used as features in CBRS is divided into two categories: the first is the default content about relevant items, such as the name of a song, the cast of a movie, and the author of a book. The second one is user-extracted features like ratings, comments, and comment tags. To extract more information and improve the quality of recommendations, De Gemmis et al. (2015) mention that Natural Language Processing and Semantic technologies have become very popular. Also, Jannach et al. (2010) support this by saying that using semantics is a particularly inventive line for CBRS.

In this research, we will build a content-based recommender system based on comment tags to recommend movies to users. We aim to answer to what extent a CBRS that uses only user-extracted features can produce the same results as a CBRS that uses both user-extracted and default content. The purpose of this question is to determine, without any fundamental knowledge about the movie, how much we can predict what users will like. Such a system could be used to provide "movie-agnostic" recommendations, meaning that the recommendations are invariant to the movies' genre, cast, and other default properties'. This kind of CBRS could recommend similar movies, not based on the similarity between movie metadata but instead on the similarity between the comments that movies have. This could broaden the scope of the recommendations.

2.5 *Scientific Relevance*

Most of the features that CBRSs are built on are text-based. Textual data could be any information about an item. In most cases, the features of an item are extracted terms from the item's description (Musto et al., 2022). Ziani et al. (2017) use comments, Farnadi et al. (2018) use metadata of movies such as genres. Regarding movies, we use comment tags as textual information. Bogers (2018) define tags as "an information classification paradigm where the users themselves are given the power to describe and categorize content for their own purposes." In other words, comment tags are textual information that is subjective and, in the context

of recommender systems, are considered user-extracted features. This is also one of the motivations in this research to use only the comment tags as features for a movie CBRS.

In CBRS, traditional methods based on term counting were not able to capture the semantic relationship between words. Semantics is identified as "meaning and meaningful use of data" (Woods, 1975). In other words, data semantics are used to develop meaningful mappings that relate items in the data. To develop meaningful mappings from item descriptions to potential users, semantic-aware recommender systems emerged (Boratto et al., 2017). We will use different semantic approaches to comment tags to potentially improve the performance of the recommender system.

2.6 *Research Strategy*

In this research, we use the MovieLens 25M dataset by (Harper & Konstan, 2015), which includes information about the metadata of the movies, ratings of the users, comment tags, and genome tags that are generated by a machine learning model created by MovieLens to be used as a reliable source for building the CBRS system (Vig et al., 2012). We built four different models to summarize our results. There will be two baselines to compare our approach to building a CBRS system. We compare our baseline models with the models that we created for CBRSs by using two different sources of knowledge bases for augmenting comment tags. We examine the evaluation process of the relevant research and give our findings based on accuracy, precision, recall, and f1 score.

3 RELATED WORKS

There are a number of studies that explore content-based movie recommender systems. These studies provide information on how to develop, optimize, and evaluate CBRS and their effects. These studies serve as the basis for this research to create the CBRS.

Pazzani and Billsus (2007) identify a recommender system as an interface that lets users build a representation of their personal preferences and then uses this to recommend similar items.

There are mainly three techniques to develop a recommender system: content-based, collaborative filtering, and hybrid approaches (Adomavicius & Tuzhilin, 2005). Content-based recommender systems (CBRS) use the features of the items based on the information and description based on default product properties such as size, cost, and technical specifications to recommend an item to a user based only on what the user has previously consumed (Pazzani & Billsus, 2007). Colloborative filtering makes predic-

tions based on the interests of a user compared to the consuming habits and preferences of a large dataset of similar users (Goldberg et al., 1992). Lastly, hybrid methods are the combination of colloborative filtering and content-based approaches, using the strengths of both methods to build large-scale, robust models (Burke, 2002).

Starting in the 1990s, several recommendation systems were introduced. In 1997, Terveen et al. (1997) presented a recommendation system called "PHOAKS" that is based on collaborative filtering, filters information from e-messages, and gives web resource and FAQ recommendations for the first time. Belkin and Croft (1992) discuss a survey about CBRSs and the information about items used by the approaches of TF-IDF and Rocchio's method.

The basic process performed by a CBRS consists of matching up the attributes of a user profile, in which preferences and interests are stored, with the attributes of a content object (item), in order to recommend to the user new interesting items (Lops et al., 2011).

3.1 *Advantages & Disadvantages of CBRS*

When compared to collobrative filtering, (Lops et al., 2011) line up the advantages of CBRS in three main sections, which are user independence, transparency, and recommending new items. User independence is an advantage of a CBRS system since the model created relies on item descriptions instead of relying on the preferences of different users (Pazzani & Billsus, 2007). Only the selected user's history is used by the model to check for relevant items in that user's profile. In this study, we build a generalizable machine learning model using the user's history as an input. We use the same users in the training set as in the validation and test sets.

Musto et al. (2022) list the limitations of content-based techniques as over-specialization, new users, and limited content analysis. Limited content analysis means a lack of suitable suggestions if there are no descriptive features. Lops et al. (2011) puts forward that there is no CBRS system that provides satisfactory recommendations if there is not enough information available about the items that are to be recommended. We also take this into consideration in our study and expect at least 100 ratings from a user. That also means that we do not have any new users in our testing dataset. We only use the users that we use in the training dataset. Later in the methodology part, we give more information about the dataset and how we use it.

Lops et al. (2011) separates the CBRS architecture into three sections: content analyzer, profile learner, and filtering component. The content analyzer's part is to convey the information obtained from the items'

information sources. In this study, we change the description of items on different models and examine how this results in recommendations.

3.2 *CBRS Architecture*

Lops et al. (2011) separates the CBRS architecture into three sections: content analyzer, profile learner, and filtering component. The content analyzer's part is to convey the information obtained from the items' information sources. In this study, we change the description of items on different models and examine how this results in recommendations.

The profile learner part is the second section of the architecture in (Lops et al., 2011), which creates a user profile by using the history of the user's interaction with items based on what they like and do not like. It provides information on how relevant the user profile is to the items that are to be recommended. (Pazzani & Billsus, 2007) separates the creation of the user profiles into two categories that are used by most of the recommender systems. The first one is the model created by the representation of the user's preferences. This model uses the description of items to capture the interest of the user and creates a user profile based on this description. The second one is the information that is gathered by the user's interaction with the items, this helps to create a history of what the user is interested, and helps to recommend similar items based on the history. To give an example, likes on social media, ratings, and purchase history can be used for this approach. Lops et al. (2011) imply that it is necessary to create a user profile in order to give recommendations to that specific user for CBRS. The user history we use in this research is the rankings they give to the movies, which we assume as a ground truth to build our CBRS models.

The third part of the CBRS architecture based on Lops et al. (2011) is the filtering component. It gives a classification result based on the user profile and item description based on their relevance. The more relevant results are the ones that are recommended to the specific user. According to Pazzani and Billsus (2007), using a history of the user's interaction is a method for classification tasks. We use the history of the user's choices in a classification task by dividing them into binary categories: positive reviews and negative reviews.

3.3 *Semantic Information*

Musto et al. (2022) describe semantics in CBRS with two approaches: endogenous and exogenous. While endogenous systems determine the meaning of a word by evaluating its use in large text corpora, exogenous semantics uses structured encodings derived from external knowledge.

The difference between the two approaches is that endogenous semantics uses only the words that the data contains, so the information is derived internally. On the other hand, exogenous systems have background knowledge encoded through an external knowledge base, so the information can be derived from outside knowledge. Both approaches have produced successful results. Word2Vec (Mikolov, Sutskever, et al., 2013) and Doc2Vec (Le & Mikolov, 2014) can be given as examples of endogenous semantics. There are several knowledge bases to give as an example for exogenous semantics, such as WordNet (Miller, 1995), BabelNet (Navigli & Ponzetto, 2012), and Wikidata (Vrandečić & Krötzsch, 2014).

3.3.1 *Endogenous Approach*

In 2013, the Word2vec model was introduced by Mikolov, Chen, et al. (2013). Word2Vec is used for calculating word vectors. G. Liu and Wu (2019) describe a word vector as a feature of a word that represents the word. To create the word vectors, unsupervised corpora are used. The main purpose of creating a word vector model is to construct a set of features for each word.

Doc2Vec, an extension of Word2Vec first introduced by Le and Mikolov (2014), is used to produce vectors for sentences, paragraphs, and entire texts. Doc2Vec is an unsupervised algorithm that produces fixed-length vectors for each item that describe the relationship between the items. In this study, it is important to have a distributed representation of the movies in a fixed dimension of vector space to create a machine learning model. As Doc2Vec is a neural network-driven approach, it captures the semantic relationships in large text documents. The Doc2Vec model works well to mine the features better than traditional language models, such as Nandi et al. (2018) find that Doc2Vec outperforms the topic modeling techniques LDA and LSA in terms of accuracy. G. Liu and Wu (2019) use Doc2Vec with the collaborative filtering method and used Doc2Vec in order to extract semantics and grammar. On the other hand, Singla et al. (2020) use Doc2Vec with tf-idf as a hybrid method. As content-based filtering was also a part of the research, they aimed to provide personalized recommendations.

We also use the Doc2Vec algorithm to vectorize the corpora we created for movies. The choice comes from the need to calculate similarities between the movies based on their corpuses instead of getting individual word features. Doc2Vec makes comparing larger pieces of text more convenient than Word2Vec.

3.3.2 *Exogenous Approach*

In order to make accurate recommendations in the system for content-based filtering, there should be sufficient content. In their research, Alharthi and Inkpen (2015) mention that K-nearest neighbors in content-based filtering perform better than collaborative filtering when Wordnet is used.

Wordnet is used as a large lexical database. In 1998, Wordnet was first introduced as a combination of traditional lexicographic information and modern computing (Miller, 1998). According to Fellbaum (1998), Wordnet consists of synsets. Synsets are described as a group of words, each of which represents a separate idea. Synsets are linked to each other based on semantic and lexical similarities. In this study, we use synsets to augment comment tags.

In the case of Wikidata, with the loads of data collected by Wikipedia since 2001, it has been decided to manage this data on a global scale. As an open source built by a global community, Wikidata is used with different knowledge bases, such as DBpedia, for content-based recommender systems (Rosati et al., 2016). These knowledge bases contain different relationships between the words.

The main reason to have different knowledge bases for the exogenous approach is that the semantic relationship between words can be defined in different ways. (K. Liu et al., 2017) points out that Wordnet is a linguistic knowledge base that focuses on the relationship between the words on a lexical level, whereas Wikidata is a general-purpose knowledge base that covers many aspects of human knowledge. In Semeraro, Lops, et al. (2009), a different approach to exploiting Wikipedia in the content analysis step is used. In order to accomplish a more accurate content analysis, they used a knowledge infusion process in content-based recommender systems. With the help of encyclopedic knowledge, Wikipedia is modeled using Semantic Vectors, based on the WordSpace model (Sahlgren, 2006), a vector space whose points are used to represent semantic concepts. It is aimed at producing new features in the recommendation process by exploiting the relationships between words and then using the activation algorithm. In this research, we compare these two exogenous approaches in an attempt to answer RQ1.

In this research, we acknowledge the limitation of having enough content available for Doc2Vec to create a CBRS, and we try to provide more information about the item descriptions by augmenting comment tags by using Wordnet and Wikidata knowledge bases.

3.4 *CBRS Models and Evaluation Methods*

Content-based recommender systems are systems to create recommendations for different domains out of text (Semeraro, Basile, et al., 2009). There are several different algorithms used for CBRS. Machine learning techniques, generally used in the task of inducing content-based profiles, are especially preferred for text categorization (Sebastiani, 2002).

One of them is a decision tree. They have labels for internal nodes that receive weight in accordance with the terms, and their leaves have labels for categories. They learned by recursively partitioning the text documents into subgroups until they only had single classes. Decision trees are used in the Syskill & Webert (Pazzani et al., 1996) recommender system. Another algorithm used for the same purpose is nearest neighbor. The "nearest neighbor" or "k-nearest neighbors" labels the class for an unclassified item, and with the similarity function, the class label is derived. Daily Learner (Billsus & Pazzani, 2000) and Quickstep (Middleton et al., 2004) use the nearest neighbor algorithm to create a model of the users' interests by associating the semantic annotations of papers within the ontology. We also used a decision tree algorithm called XGBoost as the main classifier of our models, and as side models, we use KNN and Random Forest algorithms to evaluate the performance of XGBoost.

Portugal et al. (2018) mentions that there are a large number of studies using root mean squared error (RMSE) and mean absolute error (MSE) due to their simplicity and effectiveness. Additionally, many machine learning algorithms used in the development of recommender systems performed well in the survey that they conducted in terms of precision, recall, and f score. We use accuracy, precision, recall, and f score as our main evaluation methods since we chose to model a binary classification task for our algorithms.

Precision, recall, and accuracy measures make up the majority of the evaluation for content-based text classifiers. Precision is defined as the number of relevant selected items divided by the number of selected items. Recall is defined as the number of relevant items selected divided by the total number of relevant items available. For the evaluation of recommender systems, they have been used in (Billsus, Pazzani, et al., 1998) and (Basu et al., 1998). F1, which is also used as an evaluation method in recommender systems (Sarwar et al., 2000), is a combination of precision and recall.

Burke et al. (2011), used prediction accuracy and precision of the recommendation lists. Even though both of these measures were found to be insufficient over time, we do not provide a list of recommended items in our experimental study. Instead, we use the user history as a

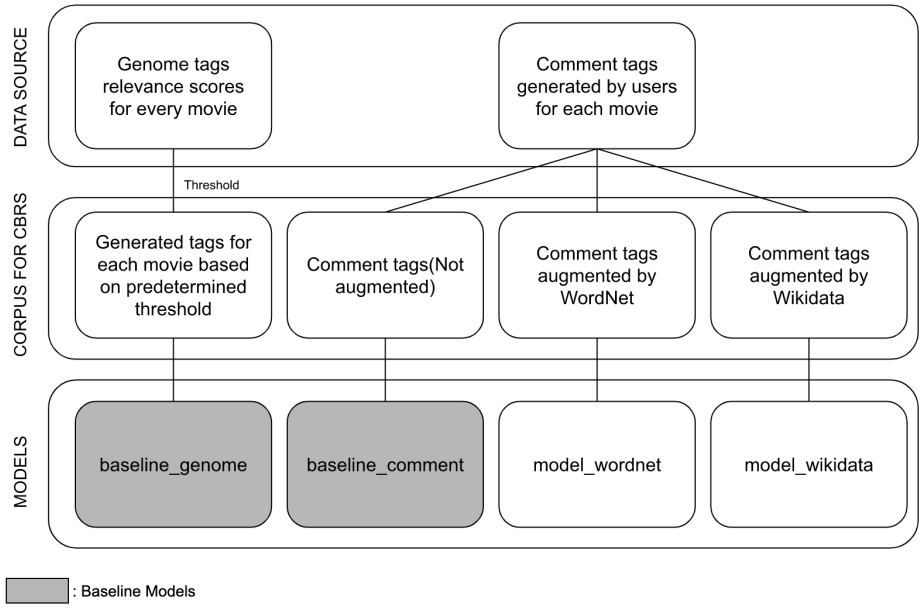

Figure 1: Figure showing which database is used to create corpus of the movies and modeling.

test set and take the labeled movies by users as ground truth. Therefore, accuracy and precision are suited for our experimental setup. Portugal et al. (2018) mention that there are a large number of studies using root mean squared error (RMSE) and mean absolute error (MSE) due to their simplicity and effectiveness. Also, in the survey that is done by them, many machine learning algorithms to build recommender systems performed well in precision, recall, and F-measure.

4 METHODOLOGY

In this study, we aim to compare different content-based recommender system models by using comment tags from users. We aim to augment comment tags by using Wordnet and Wikidata to improve the accuracy of the system.

In total, we will compare four approaches, which were created by using genome tags, comment tags, comment tags augmented by Wordnet, and comment tags augmented by Wikidata. Figure 1 shows four different models trained using different datasets, which consist of the two baselines, wikidata, and wordnet models. The methodology outlined is a guide for the data collection, preprocessing, model construction, and evaluation processes.

The model baseline_genome will use genome tag relevance scores to collect relevant information about each movie. These genome tags were provided by MovieLens (Vig et al., 2012). The genome tag set was created by a machine learning algorithm using content from user-contributed tags, reviews, and ratings. The genome tag set contains all meta-data information about movies. This gives us the chance to build CBRS with all of the metadata available about the movies. On the other hand, our experimental models only use comment tags as an information source. As a second baseline, the model baseline_comment will train the recommender system using only comment tags. In the model_wordnet and the model_wikidata, the content of the movies will be directly linked with comment tag data. Comment tags will be augmented using WordNet and Wikidata knowledge bases, respectively.

The purpose of this research is to build a reliable CBRS by utilizing user-generated comment tags. We first build a CBRS only using the user-generated comment tags in their original form, and after that, we try to improve the semantic relationship between the items by using different augmentation methods from the comment tags. We use different knowledge bases to explore how different semantic relations between words affect information retrieval between items. We purposefully excluded information derived directly from movie metadata. The metadata in Movielens dataset refers to the details about the movies, which are the characteristics of the movie. It is excluded since we are investigating if it is possible to create a robust recommender system using only generated tags for the items. The purpose of this is to construct a CBRS without relying on such information, and to improve its accuracy if we do not have this information. The metadata we exclude for this research covers genres, release dates, cast, and plot summaries, which are proven to be effective ways to retrieve information about the movies. We incorporate the comment tags as a primary source of information to explore the potential of user-generated features. We leveraged the user-generated features by augmenting them with more features using WordNet or Wikidata. We evaluate the impact of different augmentation methods on overall accuracy.

4.1 *Data Collection*

We use the MovieLens 25M dataset for this study (Harper & Konstan, 2015). The dataset has been collected by the GroupLens Research Project at the University of Minnesota and is widely used for movie recommendation research (Zhou et al., 2017), (Ali et al., 2018). The dataset contains 25 million ratings, 1 million comment tags, genome tags, and metadata information about 59047 movies, 45251 of which have been commented on by a user.

userId	title	tag	genres
131341	Pulp Fiction 1994	amazing	Comedy | Crime | Drama | Thriller
70092	Invisible Invaders 1959	BD-R	Horror | Sci-Fi
155752	Interstellar 2014	space travel	Sci-Fi | IMAX
68120	Chorus, The *Choristes, Les* 2004	France	Drama
84533	Splice 2009	Sarah Polley	Horror | Sci-Fi | Thriller
157590	Star Wars: Episode IV - A New Hope 1977	space adventure	Action | Adventure | Sci-Fi
11602	Traffic *Trafic* 1971	Jacques Tati	Comedy
126013	Due Date 2010	Michelle Monaghan	Comedy
31047	Cleaner 2007	SINGLE PARENTS	Crime | Thriller
124601	To Catch a Thief 1955	heist film	Crime | Mystery | Romance | Thriller
39983	Addams Family, The 1991	supernatural	Children | Comedy | Fantasy
6550	Private Life of Sherlock Holmes, The 1970	sherlock holmes	Comedy | Drama | Mystery
6550	Haunted Palace, The 1963	mutations	Horror
105728	Honeymoon 2014	woman director	Horror
33844	Dunkirk 2017	evacuation	Action | Drama | Thriller | War

Table 1: Random movie samples from user comment tags

The provided data makes the dataset a very rich source for building a recommendation system. Our study will consider movies that have at least one comment. Hence, we investigate the impact of comment tags on a CBRS system.

The dataset comes in CSV format and contains two tag files. The first one is the tags.csv dataset, which contains comment tags applied to movies by users. "The meaning, value, and purpose of a particular tag are determined by each user," according to the source (Harper & Konstan, 2015). We achieve the main approach of the research question by using the tags.csv file, which includes all of the comment tags from the users. We use these comment tags as input to a CBRS model. This is our first baseline to compare with augmented models. Also, we augment these comment tags by using Wordnet and Wikidata to increase their accuracy.

In the tags.csv dataset, 45251 users have contributed 1093360 tags. Each user has at least one tag and a maximum of 183356. The average tag count for each user is 75. In terms of interquartile ranges (IQR), a 0.25 quantile equals 2 tag counts, and a 0.75 quantile equals 20 tag counts. There are 4613 users with one or two tags. There are 6325 users with tag counts between 0.25 and 0.75 IQR and 3654 users with tag counts greater than 0.75 IQR. We show random comment tag samples from users in Table 1.

The second tag file, namely the genome_tags.csv dataset, contains 1128 tags. Every tag in each movie has a relevance score. The genome tag data is a dense matrix of relevance scores for each tag in each movie. The advantage of a dense matrix is that we can mine the same number of tags for each movie. We mine movie tags using a relevance score threshold. We assess the relevancy of each movie across all genome tags. Table 2 displays an example of genome tags with relevance scores derived from random movies. We create a second baseline model using genome

title	relevance	tag	genres
Ace Ventura: When Nature Calls (1995)	0.92075	sequels	Comedy
Personal Shopper (2016)	0.718	loneliness	Drama | Thriller
Kill Bill: Vol. 2 (2004)	0.70025	gangsters	Action | Drama | Thriller
Delivery Man (2013)	0.579	mentor	Comedy
Futureworld (1976)	0.64725	technology	Sci-Fi | Thriller
Jurassic Park (1993)	0.54275	clever	Action | Adventure | Sci-Fi | Thriller
Born Free (1966)	0.917	animal movie	Adventure | Children | Drama
The Hunger Games: Mockingjay - Part 1 (2014)	0.7085	science fiction	Adventure | Sci-Fi | Thriller
Winter War (Talvisota) (1989)	0.989	war	Drama | War
Last Tango in Paris (Ultimo tango a Parigi) (1972)	0.52475	life & death	Drama | Romance
Lost Highway (1997)	0.975	strange	Crime | Drama | Fantasy | Film-Noir | Mystery | Romance
Good Morning (Ohayô) (1959)	0.8405	imdb top 250	Comedy
Never Let Me Go (2010)	0.53475	slow paced	Drama | Romance | Sci-Fi
Eight Legged Freaks (2002)	0.50825	low budget	Action | Comedy | Horror | Sci-Fi
Twilight (2008)	0.5335	bad plot	Drama | Fantasy | Romance | Thriller

Table 2: Examples of genome tags with relevance scores higher than 0.50

tags to investigate the accuracy of a CBRS system that uses all available information.

The Movielens dataset also includes ratings.csv and movies.csv files. In the ratings.csv files, we have ratings from the user for the movies. For this study, we transformed the original movie ratings into binary encodings. The ratings in the dataset are given from 0 to 5, incrementing by 0.5. We utilized the median rating as a threshold to categorize ratings into two classes: 1 (positive review) and 0 (negative review). A rating equal to or greater than the median value is considered a positive review for users. The main reason to create binary encodings is to use a simple methodology for our CBRS. Also, in this study, we do not try to determine how much a user likes a movie, instead, we only try to recommend or not recommend items. We apply that by dividing the rating dataset into train and test sets 80% and 20%, respectively. We evaluate the performance by comparing the label we get from the test set with the actual truth on a binary level. This allows us to draw confusion matrices easily, interpret the results, and take action on hyperparameter tuning.

The movies.csv contains the information of movie ID, title, and genres. We do not use titles or genres in this study. We only use titles for visualisation, not for model development. Also, there is a links.csv dataset that contains the information about the movie ID and IMDB links to get more information about the movie. This file is also not part of this study.

4.2 *Model Development*

In this study, we use a structured process to build our CBRS and evaluation processes. We use separate classes to divide the key steps, which include data loading, preprocessing, which contains denormalization and tokenization steps, augmentation processes for model_wordnet and model_wikidata, vectorizing by using Doc2Vec, train-test split, user vector

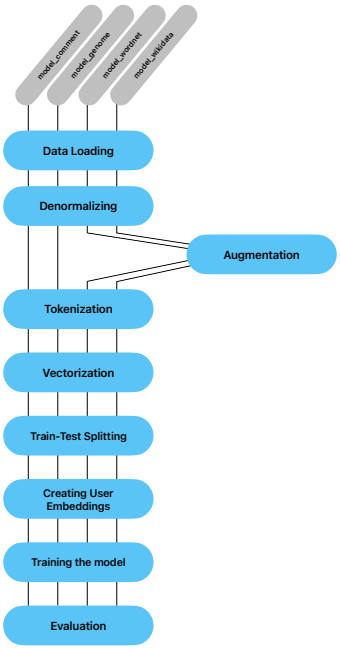

Figure 2: Figure showing the architecture of the experimental setup.

generation, model training, and evaluation. Figure 2 shows the model architecture and which models use which steps.

After data loading, we start preprocessing by denorming the comment tags and genome tags. By denorming, we mean that we created a corpus for each movie from relevant data sources for the models. The second step of the preprocessing is tokenization. Since we use English words for both tags, we use English stop words to tokenize the corpus we have for each model. Stop words mean the unimportant words that are used in English, such as 'the', 'an', 'a'. It is important to remove those words since they might affect the similarity between the movies. We try to avoid bias just because there are many similar stop words in irrelevant movie corpuses. We convert every letter to lower case, and then we remove all of the stop words from the corpus of each dataset created for models.

4.2.1 *Augmentation Process*

We use the augmentation process for model_wordnet and model_wikidata in between the processes of denormalization and tokenization. It is the exogenous approach that we use for the models wikidata and wordnet. For the model_wordnet we use synonyms of comment tags. A group of synonyms with a similar meaning is known as a synset. We iterate over

each synonym in a synset, and if it isn't already in the list of synonyms, we add the word. We only use synonyms because we use WordNet as a lexical database for this study. One of the purposes of this study is to compare the models that are augmented by lexical semantic relationships and non-lexical semantic relationships. Therefore, in this study, Wordnet stands for the lexical semantic knowledge base that we use for augmentation. For the model_wikidata, for each tag, we get the first 250 characters of the summary from the Wikipedia page if it exists. Then, we use the NLP method in the SciPy library. This method gives us the entities by using the summary that we get and tokenizing it to ensure we only have the entities. Lastly, we use a limitation of augmented words for each comment tag in both models.

While determining the limit, there is a trade-off between computational cost and getting enough new information. Therefore, we use the limitation technique as a hyperparameter and divide it into three categories: low limit (1 word), moderate limit (3 words), and high limit (10 words). We use the low limit to test whether a small amount of augmentation provides any benefit. We assume the moderate limit could be an optimal point that offers additional information without overcomplicating the model. In the high limit, we test whether a large amount of augmentation significantly improves performance. We analyzed the results only by augmenting the Wordnet knowledge base since Wikidata is already computationally expensive, and we implemented the same results on Wikidata to get an equal amount of information. We augment three words for every comment tag used in the corpus and add them together without removing the original comment tags.

4.2.2 *Vectorization & Creating User Embeddings*

After tokenization, we use Doc2Vec to create vectorized movie embeddings for the machine learning stage. It is important to vectorize to capture the similarities between the movies based on the corpus they have, and we use those similarities to predict future recommendations. This vectorization allows us to capture semantic relationships between the movies, and different inputs to the models give us different relations. As mentioned in the related works part, it is the endogenous approach that we use for all the models. All models use Doc2Vec to vectorize the corpus of movies that we gather for the models. This dataset provides us with the relationship between all movies in our dataset. We use these vectors to create user embeddings, which are necessary to build a recommendation engine.

The hyperparameters we use for Doc2VEc to optimize the performance of the model are vector size, distributed memory (dm), number of epochs, and learning rate. Vector size is one of the key hyperparameters that we

consider in this project. The reason for this is that to have distinct representations for each movie, we need enough vector size to represent it. On the other hand, increasing vector size causes more expensive computations for both Doc2Vec and XGBoost models. Vector size tuning attempts are mainly based on the accuracy of the XGBoost model and computational costs. We chose to continue with 100 vectors for each movie.

The second important hyperparameter is the distributed memory parameter. This parameter has two options. The first one uses CBOW (Continuous Bag Of Words), and the second one uses DBOW (Distributed Bag Of Words). DBOW is not capturing the order of the words inside the given corpora, while CBOW tries to capture the relationship in the order of the words as well. In this study, the order is not important since we use comment tags and genome tags instead of sentences. This ensures that similar words among the movies can be represented more independently without consideration of the order, allowing us to capture more similarities between movies that need to be similar.

We optimize the number of epochs and learning rate based on the evaluation metrics of XGBoost. Number of epochs is chosen 50 and the learning rate (alpha) is chosen at 0.025 after several tuning attempts, and the minimum learning rate is 0.00025.

To evaluate the performance of the Doc2Vec models before the accuracy of the XGBoost model, we use visual mappings and similarity scores between movies. We try to capture relationships between similar movies, like animations. Also, we check whether movie series are similar to each other. These are the two methods to evaluate the performance of the unsupervised learning algorithm before the implementation of the classification model.

For creating user embeddings, we first split the rating dataset, which contains userId, movieId, and binary ratings, into train (70%), validation (15%), and test (15%) datasets on users. By dividing the dataset by users, we ensure that our train, validation, and test datasets have the same users. Since we are creating a CBRS, we try to recommend movies to a user by looking at their own history. These models contain all of the users to make the algorithm more generalizable. To create the features from the train dataset, we need the information for each movie and each user. For each movie, we have 50 vectors that will be used as features of the train dataset. For each user, we use all of the movies that are in the train set, and we take the weighted average for each vector by using ratings from 0 to 5. At the end, there are 50 features for each user that contain information about their movie history and the respective ratings given. We combine the user embeddings and movie embeddings together to get a train set with 100

features. We use the same user embeddings in validation and test sets, which only contain user information from the train set.

4.2.3 *Models*

We train our datasets on an XGBoost model, which is a gradient boosting algorithm. And to compare the results with the other models, we train and evaluate the datasets, including k-nearest neighbors (KNN) and random forest (RF).

The key hyperparameters that we use to adjust the performance are learning rate, number of estimators, maximum depth. We use grid search with 5 cross validation sets to find the best parameters.

We use logarithmic loss as an evaluation metric for the XGBooost algorithm. Logarithmic loss is a popular evaluation metric and gives us the probability of a predicted label between 0 and 1. We divided our predictions from the 0.5 threshold into binary classes. We have given a place in the discussion section to explain how the prediction probabilities from logarithmic loss evaluation can be used based on the expectation of precision and recall values.

We evaluate the hyperparameters of the XGBoost model by using ROC-AUC scores.

5 RESULTS

In this section, we talk about the results of the four models. We first compare the four models and interpret the main result. Then we examine how we evaluate the performance of the vectorizing step while using Doc2Vec. Also, we show how we use ROC AUC curves in hyperparameter tuning and evaluating the performance of XGBoost.

We use accuracy, precision, recall, f_score as evaluation metrics in this study. We use accuracy as a main performance metric to compare the models against each other. For XGBoost hyperparameter tuning, we also consider weighted f score calculation to evaluate the performance of the model and make sure that we calculate the slight imbalance between classes.

5.1 *Main Results*

As shown in the Table 3 the baseline_genome gives 73% accuracy using all available information, and the baseline_comment gives 71% accuracy using users' comment tags. The model enhanced with augmented comment tags by wikidata and wordnet give 72% and 69% accuracy, respectively, which

is 2% below the genome results on average. This slight decrease in the accuracy score shows us that the corpuses used for data augmentation are not improving the quality of the comment tags in terms of being a feature representation for movies. We concluded that the new corpuses are not significantly better for the classification of the comments. As the baseline model gives higher accuracy, it can be stated that only using user-extracted features as an input is not enough to improve the content-based recommendation.

	Accuracy	Precision (Macro)	Recall (Macro)	F1 Score (Macro)	F1 Score (Weighted)
baseline_comment	0.7136	0.7127	0.7101	0.7107	0.7127
baseline_genome	0.7341	0.7326	0.7300	0.7308	0.7333
model_wordnet	0.6954	0.6951	0.6946	0.6947	0.6952
model_wikidata	0.7229	0.7219	0.7202	0.7207	0.7224

Table 3: Test results of the models.

We compare the four XGBoost models to answer RQ1 in Table 3. In our comparison, we see that, as an interesting finding, model_wordnet falls below baseline_comment in our evaluation metrics. We see that baseline_comment and augmented models do not reach the accuracy and f1-score scores obtained with model_genome. Apart from these, we have reached the conclusion that precision and recall metrics are almost at the same score for the models.

We used accuracy, one of the performance metrics we found, for two-tailed paired t-tests. Two-tailed paired t-tests were used to assess the statistical significance of any changes in performance between two predictors (Mitchell 1997). To do the t-test, we use the cross-validation results of the models. With the resulting mean and standard deviation, we tested whether the scores of the two models came from the same population.

When we look at the confusion matrices of the models in Figure 3, we observe that the movies which are relevant (class 1) to a user are predicted at a better rate than the movies that are relevant (class 0). Additionally, it demonstrates that in each model, our rate of false positives is higher than our rate of false negatives. This indicates that our Type 1 error is larger than our Type 2 error, and furthermore, we find that models have a tendency to propose movies even when they are irrelevant.

5.2 *Doc2Vec evaluation of the models*

When we look at the Figures 4, 5, 6 and 7, which are the UMAPs generated by using umap library in Python, how the embedding spaces are parameterized is similar to the final results of the recommendation system in Table 3. In this experimentation pipeline, the main difference between these models comes from the fact that the input data semantically captures

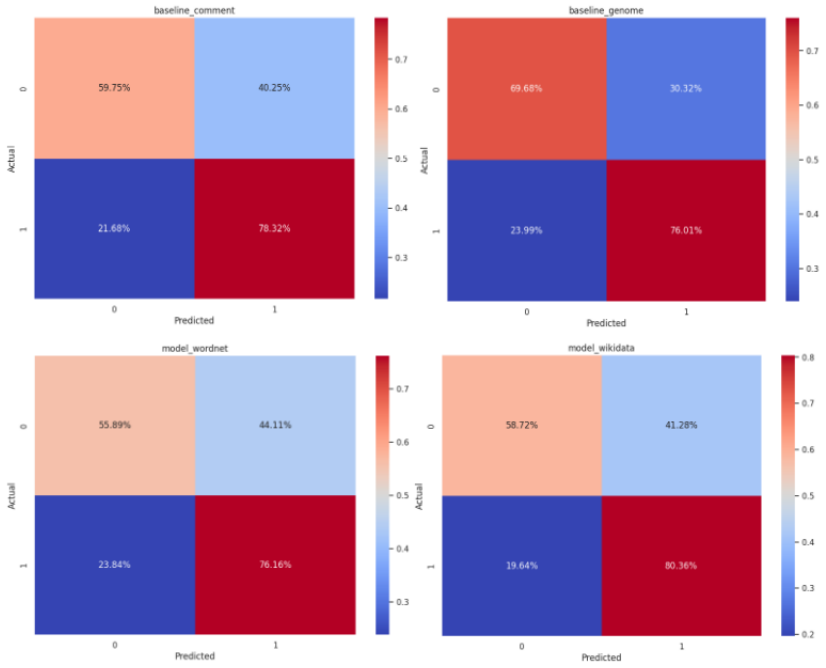

Figure 3: Confusion matrix of all models.

the similarity of the movies. We evaluate the Doc2Vec hyperparameters based on the genome tags, as we use these tags as ground truth. At this point, we can clearly see that the popular movies shown in model_genome are positioned closer than other UMAPs. The basis of the input data for the other three models comes from the comment tags. When we compare model_wikidata and model_wordnet, we observe that the model_wordnet is more dispersed. This can be interpreted as meaning that the words we augment with the wordnet model are reflected in the data as more noise than information. This may be why model_wordnet further degrades the performance of the baseline model while model_wikidata improves accuracy.

When we use both the performances in the Table 3 and the UMAPs to answer RQ2, we can say that model_wikidata performs better than model_wordnet. However, UMAP images indicate that vectorized movies need to be revalidated, this result may be biased.

5.3 *Hyperparameter Tuning of XGBoost Models*

We used ROC-AUC curves for hyperparameter tuning. The new curves created by the models as a result of our tuning are shown in Figures 8, 9, 10

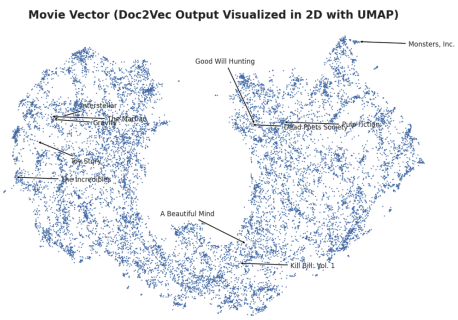

Figure 4: UMAP of model_genome

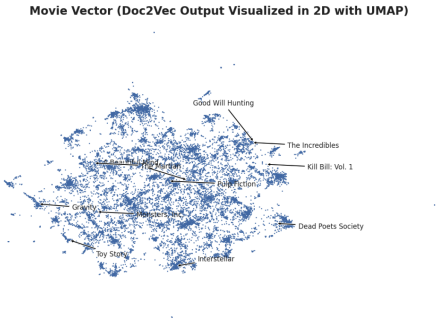

Figure 5: UMAP of model_comment

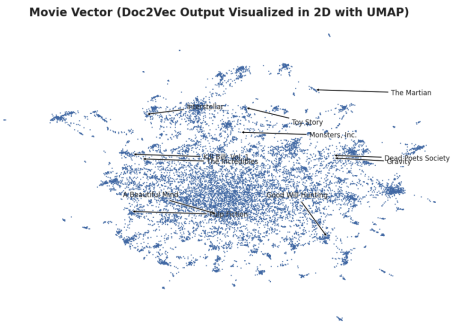

Figure 6: UMAP of model_wordnet

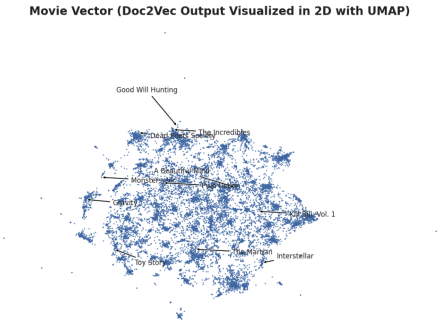

Figure 7: UMAP of model_wikidata

and 11 in orange. We see improvements on every model. Hyperparameters were most effective on model_wordnet.

5.4 *Comparison of the XGBoost model with Side Models*

	XGBoost	**KNN**	**Random Forest**
baseline_comment	0.7136	0.6243	0.6800
baseline_genome	0.7341	0.6305	0.6442
model_wordnet	0.6954	0.6246	0.6395
model_wikidata	0.7229	0.6284	0.6557

Table 4: Accuracy results of the main model and the side models

We use k-nearest neighbors and a random forest classifier to compare with XGBoost to answer RQ3. In terms of accuracy, XGBoost is the best of all of them. The Random Forest classifier did well in the baseline_comment, which was an interesting finding for us. Cross validation was not carried

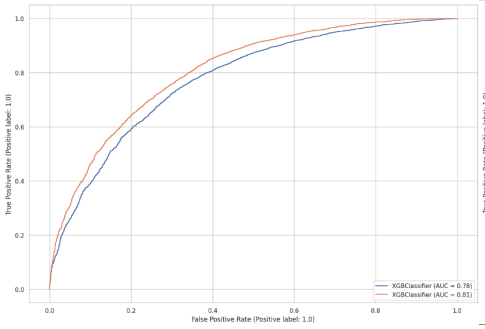

Figure 8: Hyperparameter optimization on model_genome

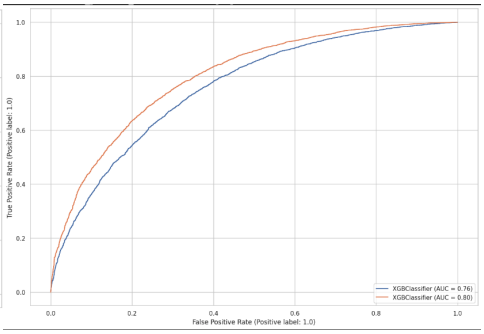

Figure 9: Hyperparameter optimization on model_comment

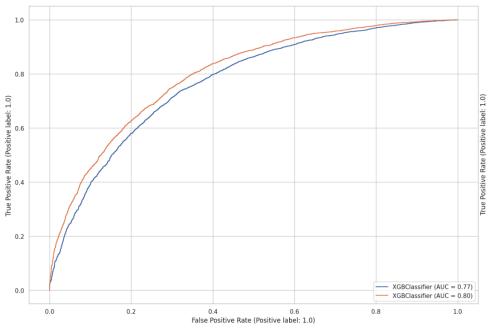

Figure 10: Hyperparameter optimization on model_wordnet

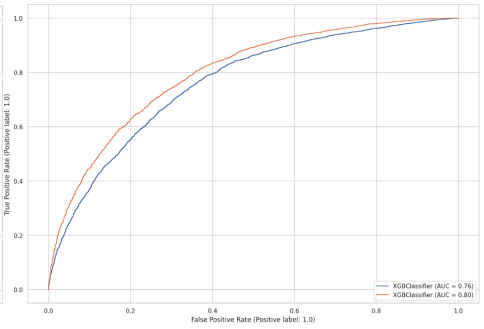

Figure 11: Hyperparameter optimization on model_wikidata

out on side models, so this finding could change when cross validation is done. In the same way, we can see that the baseline_genome dataset comes first in the XGBoost model, but in the Random Forest method, the baseline_comment and model_wikidata datasets do better than baseline_genome. When analyzing RQ3, the results show that decision tree models are still better than KNN models.

6 DISCUSSION

As stated in the result section above, we compare the model_genome, model_comment, model_wikidata and model_wordnet based on accuracy and precision, recall and f1 score.

Exogenous features that the augmented corpuses add to the comment tag data are not statistically significant when compared to non-augmented data, as stated in the RQ1. In addition, considering the RQ2, neither of the knowledge bases used improved the accuracy scores compared to baseline

comment tags. Comparing both Wikidata and Wordnet, we can justify that neither of the corpuses were enough to surpass the genome dataset that is not augmented. A comparison of the two different corpuses showed this much difference between them. Even though this is not a significant improvement, it can be said that the wikidata corpus performs slightly better than the wordnet corpus. We can conclude that the user-extracted features do not have a high impact on movie recommendations. It should be noted that content-based recommender systems are known for over-specialization, which prevents the recommender system from interacting with others that have similar user profiles.

In terms of accuracy, genome tags produced by a machine-learning model are not significantly different from comment tags. Having said that, using only user-extracted features does not make a significant difference from all available meta information. This shows us that user-generated comment tags for the movies are enough to provide as accurate recommendations as using all meta information.

Nevertheless, even though the results show no significant improvement across different setups, the accuracy results did not go below 50% in any situation, so this provides proof of the generalizability of the proposed model. We can also adjust the threshold of the generalizable model we have according to the appropriate business cases. For example, if the cost of making a recommendation to a user is high and we want to be sure what to recommend, we need to choose a high-precision model. In such a case, we can increase the binary classification threshold from 0.5 in the logarithmic loss function we use. In our studies, when we increase the threshold and bring the precision to 80%, accuracy becomes 65%, recall 64%. Likewise, a high recall may be important in another business case scenario. For example, nowadays, recommender systems are popular, competition is very high, and we may want our model to show the movie that a customer might like first. In this case, we can set the threshold to increase the recall, taking into account the type 1 error. When we increased the recall to 92% in the studies carried out, we saw that the accuracy was around 65% and the precision around 71%.

7 LIMITATIONS & FURTHER RESEARCH

Choosing median as a threshold for positive and negative reviews gives the movies in that area an unclear opinion. Although the ratings that are very close to the median are taken for a certain party, there may be differences that will affect the accuracy because of the human perspective. The binary classification problem can be tried without including the movies in the section that are close to the ratings in the median.

There is an imbalance in the number of comment tags in each movie. Some movies have a significantly larger number of tags than others. This imbalance may cause us not to recommend films with few comment tags. Especially in the vectorization step, similarity scores may be biased due to a lack of information about a movie that has few comment tags. Although we tried to overcome this problem with augmentation methods, the dataset remained imbalanced after augmentation.

If the comment tags used do not have a corresponding augmentation in wordnet or wikidata, we leave that word as it is. This is a limitation that needs to be examined, the effect of these words may have affected the recommender system.

For the models with comment tag augmentation, using adaptive limitations for Wordnet and Wikidata can be useful. This approach can be adaptable and links the frequency or significance of the original tag in the dataset to the number of augmented terms. By doing this, we can give tags that are more frequently used or that the model considers to be more significant greater weight.

In further research, the augmentation method can be changed to adaptive, and evaluation can be made with ordinal-encoded ratings instead of binary ratings. In this case, the evaluation metrics to be used should be changed to RMSE and MSE. Using ratings in this way can eliminate the median bias of binary ratings, which are divided into positive reviews and negative reviews based on the median. In this case, the imbalance of the rating data will be important. There may be a need to reselect the ratings that have fewer samples or use another method.

8 CONCLUSION

In conclusion this thesis aimed to investigate whether user extracted features are enough to provide as good recommendations as the features created by using all meta information. The study utilized 4 datasets, including movies, comment tags, ratings, genome tags and applied Doc2Vec, XGBoost, RandomForest, KNN Classifiers to test the impact of augmented data on model accuracy.

The results of the study indicate that the optimal accuracy of the corpus created by the models using augmentation has not significantly improved. While the model that uses augmentation by using Wordnet gives 69% of accuracy, the model that uses augmentation by using Wikidata gives 71%. Both are below the baseline created by using genome tags, which gave 73% accuracy.

Even though the accuracy of the models considered is not significantly higher than the baselines, the diversity of training and test variations in this study gives validation to the ability to generalize.

Overall, the results of the study suggest that using only comment tags to recommend a movie can be applicable. However, it is crucial to note that the study is limited to only using comment tags for content-based movie recommendation, and the results may not give the same results in different categories of recommendation systems. Further research could explore the potential of comment tags and augmentation techniques when combined with different features and recommendation techniques for more accurate and personalized results.

The findings of the study provide valuable insights for semantic relationships between items using content-based recommendation systems. In addition, the outcomes of this research can be used as a basis for further studies in the field of recommendation systems and their application in classifying movies.

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, *17*(6), 734–749.

Ahmed, S., & Jaidka, K. (2013). Protests against# delhigangrape on twitter: Analyzing india's arab spring. *eJournal of eDemocracy and Open Government*, *5*(1), 28–58.

Alharthi, H., & Inkpen, D. (2015). Content-based recommender system enriched with wordnet synsets. *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CICLing 2015, Cairo, Egypt, April 14-20, 2015, Proceedings, Part II 16*, 295–308.

Ali, S. M., Nayak, G. K., Lenka, R. K., & Barik, R. K. (2018). Movie recommendation system using genome tags and content-based filtering. In *Advances in data and information sciences* (pp. 85–94). Springer.

Basu, C., Hirsh, H., Cohen, W., et al. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Aaai/iaai*, 714–720.

Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, *35*(12), 29–38.

Billsus, D., Pazzani, M. J., et al. (1998). Learning collaborative information filters. *Icml*, *98*, 46–54.

Billsus, D., & Pazzani, M. J. (2000). User modeling for adaptive news access. *User modeling and user-adapted interaction*, *10*, 147–180.

Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, *23*(6), 520–528.

Bogers, T. (2018). Tag-based recommendation. In *Social information access* (pp. 441–479). Springer.

Boratto, L., Carta, S., Fenu, G., & Saia, R. (2017). Semantics-aware content-based recommender systems: Design and architecture guidelines. *Neurocomputing*, *254*, 79–85.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*, 331–370.

Burke, R., Felfernig, A., & Göker, M. H. (2011). Recommender systems: An overview. *Ai Magazine*, *32*(3), 13–18.

Chen, M.-H., Teng, C.-H., & Chang, P.-C. (2015). Applying artificial immune systems to collaborative filtering for movie recommendation. *Advanced Engineering Informatics*, *29*(4), 830–839.

Crespo, R. G., Martınez, O. S., Lovelle, J. M. C., Garcıa-Bustelo, B. C. P., Gayo, J. E. L., & De Pablos, P. O. (2011). Recommendation system based on user interaction data applied to intelligent electronic books. *Computers in human behavior*, *27*(4), 1445–1449.

De Gemmis, M., Lops, P., Musto, C., Narducci, F., & Semeraro, G. (2015). Semantics-aware content-based recommender systems. *Recommender systems handbook*, 119–159.

De Gemmis, M., Lops, P., Semeraro, G., & Basile, P. (2008). Integrating tags in a semantic content-based recommender. *Proceedings of the 2008 ACM conference on Recommender systems*, 163–170.

Diao, Q., Qiu, M., Wu, C.-Y., Smola, A. J., Jiang, J., & Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 193–202.

Farnadi, G., Kouki, P., Thompson, S. K., Srinivasan, S., & Getoor, L. (2018). A fairness-aware hybrid recommender system. *arXiv preprint arXiv:1809.09030*.

Fellbaum, C. (1998). *Wordnet: An electronic lexical database*. MIT press.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70.

Gomez-Uribe, C. A., & Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, *6*(4), 1–19.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, *5*(4), 1–19.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, *16*(3), 261–273.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender systems: An introduction*. Cambridge University Press.

Kartheek, M., & Sajeev, G. (2021). Building semantic based recommender system using knowledge graph embedding. *2021 sixth international conference on image information processing (ICIIP)*, *6*, 25–29.

Kenthapadi, K., Le, B., & Venkataraman, G. (2017). Personalized job recommendation system at linkedin: Practical challenges and lessons learned. *Proceedings of the eleventh ACM conference on recommender systems*, 346–347.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *International conference on machine learning*, 1188–1196.

Lee, D., & Hosanagar, K. (2014). Impact of recommender systems on sales volume and diversity.

Li, H., Cui, J., Shen, B., & Ma, J. (2016). An intelligent movie recommendation system through group-level sentiment analysis in microblogs. *Neurocomputing*, *210*, 164–173.

Liu, G., & Wu, X. (2019). Using collaborative filtering algorithms combined with doc2vec for movie recommendation. *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 1461–1464.

Liu, J., Dolan, P., & Pedersen, E. R. (2010). Personalized news recommendation based on click behavior. *Proceedings of the 15th international conference on Intelligent user interfaces*, 31–40.

Liu, K., Shi, X., & Natarajan, P. (2017). Sequential heterogeneous attribute embedding for item recommendation. *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 773–780.

Lops, P., Gemmis, M. d., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, 73–105.

Middleton, S. E., Shadbolt, N. R., & De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 54–88.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, *26*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Miller, G. A. (1998). *Wordnet: An electronic lexical database*.

Musto, C., Gemmis, M. d., Lops, P., Narducci, F., & Semeraro, G. (2022). Semantics and content-based recommendations. In *Recommender systems handbook* (pp. 251–298). Springer.

Nandi, R. N., Zaman, M. A., Al Muntasir, T., Sumit, S. H., Sourov, T., & Rahman, M. J.-U. (2018). Bangla news recommendation using doc2vec. *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, 1–5.

Navigli, R., & Ponzetto, S. P. (2012). Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, *193*, 217–250.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. *The adaptive web: methods and strategies of web personalization*, 325–341.

Pazzani, M. J., Muramatsu, J., Billsus, D., et al. (1996). Syskill & webert: Identifying interesting web sites. *AAAI/IAAI, Vol. 1*, 54–61.

Portugal, I., Alencar, P., & Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, 205–227.

Qu, W., Song, K.-S., Zhang, Y.-F., Feng, S., Wang, D.-L., & Yu, G. (2013). A novel approach based on multi-view content analysis and semi-supervised enrichment for movie recommendation. *Journal of Computer Science and Technology*, *28*(5), 776–787.

Reddy, S., Nalluri, S., Kunisetti, S., Ashok, S., & Venkatesh, B. (2019). Content-based movie recommendation system using genre correlation. In *Smart intelligent computing and applications* (pp. 391–397). Springer.

Ricci, F., Rokach, L., & Shapira, B. (2010). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Springer.

Rosati, J., Ristoski, P., Di Noia, T., Leone, R. d., & Paulheim, H. (2016). Rdf graph embeddings for content-based recommender systems. *CEUR workshop proceedings*, *1673*, 23–30.

Sahlgren, M. (2006). *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces* (Doctoral dissertation). Institutionen för lingvistik.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. *Proceedings of the 2nd ACM Conference on Electronic Commerce*, 158–167.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1–47.

Semeraro, G., Basile, P., de Gemmis, M., & Lops, P. (2009). User profiles for personalizing digital libraries. In *Handbook of research on digital libraries: Design, development, and impact* (pp. 149–158). IGI Global.

Semeraro, G., Lops, P., Basile, P., & de Gemmis, M. (2009). Knowledge infusion into content-based recommender systems. *Proceedings of the third ACM conference on Recommender systems*, 301–304.

Singla, R., Gupta, S., Gupta, A., & Vishwakarma, D. K. (2020). Flex: A content based movie recommender. *2020 International Conference for Emerging Technology (INCET)*, 1–4.

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997). Phoaks: A system for sharing recommendations. *Communications of the ACM*, *40*(3), 59–62.

Vig, J., Sen, S., & Riedl, J. (2012). The tag genome: Encoding community knowledge to support novel interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *2*(3), 1–44.

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks. In *Representation and understanding* (pp. 35–82). Elsevier.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2008). An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(2), 435–447.

Zhou, T., Chen, L., & Shen, J. (2017). Movie recommendation system employing the user-based cf in cloud computing. *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, *2*, 46–50.

Ziani, A., Azizi, N., Schwab, D., Aldwairi, M., Chekkai, N., Zenakhra, D., & Cheriguene, S. (2017). Recommender system through sentiment analysis. *2nd international conference on automatic control, telecommunications and signals*.