Expert Systems With Applications 158 (2020) 113452

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid recommender system for recommending relevant movies using an expert system

Bogdan Walek*, Vladimir Fojtik

Department of Informatics and Computers, University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 5 December 2019
Revised 7 April 2020
Accepted 12 April 2020
Available online 13 May 2020

Keywords:
Recommender system
Hybrid recommender system
Expert system
Collaborative-filtering
Content-based filtering
Movies
Rating

ABSTRACT

Currently, the Internet contains a large amount of information, which must then be filtered to determine suitability for certain users. Recommender systems are a very suitable tool for this purpose. In this paper, we propose a monolithic hybrid recommender system called Predictory, which combines a recommender module composed of a collaborative filtering system (using the SVD algorithm), a content-based system, and a fuzzy expert system. The proposed system serves to recommend suitable movies. The system works with favorite and unpopular genres of the user, while the final list of recommended movies is determined using a fuzzy expert system, which evaluates the importance of the movies. The expert system works with several parameters – average movie rating, number of ratings, and the level of similarity between already rated movies. Therefore, our system achieves better results than traditional approaches, such as collaborative filtering systems, content-based systems, and weighted hybrid systems. The system verification based on standard metrics (precision, recall, F1-measure) achieves results over 80%. The main contribution is the creation of a complex hybrid system in the area of movie recommendation, which has been verified on a group of users using the MovieLens dataset and compared with other traditional recommender systems.

1. Introduction

Currently, various recommended systems are increasing, and their main objective is to recommend the user suitable content based on various parameters.

A recommender system is an information system serving to support user decision making and recommend suitable products, information, and services in the area of e-shops, streaming services, internet dating services, and many other areas (Falk, 2019).

Recommender systems use an analysis of a specific type of data to predict user ratings for individual items. Subsequently, (based on this analysis), they create recommendations and modify the content of the displayed page so that it corresponds to the user's preferences as much as possible (Falk, 2019). This is one of the reasons why a wide range of companies and web applications have recently implemented systems analyzing users' behavior with respect to recommending the most suitable products, services, or information. The objective is, of course, to increase the sales and profits of these companies. It is primarily the area of streaming platforms and online movie rentals (generally all services distributing audio-visual works) where the recommendation of relevant products to a user plays a significant role. The main objective of all these services is that each display of a product by a user results in conversions, i.e., user action – purchasing access to watch a movie, subscription registration, etc. In addition, these services also attempt to recommend the best buyer experience possible – satisfied customers with displayed products corresponding to their taste will potentially come back and purchase something again. The most prominent players among the pioneers of recommender systems are Amazon and Netflix.

Amazon patented the first version of its recommender system as early as 2004. This system increased its profits by 29%, to 12.83 bn. dollars in the second fiscal quarter (compared with the previous year's profit of 9.9 bn. dollars) (Fortune, 2012; Lester, 2013; TAMMA Capital LLC., 2019).

Netflix implemented a recommender system in its application to decrease the number of canceled subscriptions as well as to increase the average time of user interaction with the application (i.e., number of streaming hours). The company anticipates that a combination of recommendations (Trending now, Continue Watching, Because You Watched, and others) and personalization will save up to 1 billion dollars a year, which would otherwise be invested in acquiring new customers on behalf of those who canceled (Gomez-Uribe and Hunt, 2015).

* Corresponding author.
E-mail addresses: bogdan.walek@osu.cz (B. Walek), p17077@student.osu.cz (V. Fojtik).

https://doi.org/10.1016/j.eswa.2020.113452
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

It is thus evident that these systems play an essential role in the world of online sales, and their potential (with the increasing number of people buying online) is still increasing.

The main objective of this article to propose a hybrid recommender system predictor for recommending suitable movies. This system contains a recommender module combining a collaborative filtering system, a content-based system, and a fuzzy expert system. The proposed system processes favorite and unpopular genres of the user, and the final list is created using a fuzzy expert system, which evaluates movie importance. The architecture and methodology of the proposed system are described in Section 3.

2. Related work and current state in recommender systems

Recommender systems have been developed for many years and have been applied in numerous problem domains: tourism (Logesh, Subramaniyaswamy, and Vijayakumar, 2018; Ravi and Vairavasundaram, 2016; Stanley, Lorenzi, Saldaña, and Licthnow, 2003), advertisement (Cheung, Kwok, Law, and Tsui, 2003), e-commerce (Ghani and Fano, 2002), music (Rodríguez-García, Colombo-Mendoza, Valencia-García, Lopez-Lorca, and Beydoun, 2015) and others. This work is focused on the area of recommender systems for movies.

2.1. Recommender systems for movies

Recommender systems for relevant movies are developed to suggest the most suitable movies for a user based on their preferences, favorite genres, actors, directors, and other parameters. The Netflix Movie Recommender System also adds an explanation of why given movies have been recommended. Presenting reasonable explanations helps the user understand why a given movie should be interesting for them. This approach helps increase system credibility and user loyalty (Aggarwal, 2016). Such an approach has also been implemented in MovieExplain (Symeonidis, Nanopoulos, and Manolopoulos, 2009). An interesting approach is also a proposal of suitable movies based on user emotions. The user marks three colors representing emotions (joy, anger, sadness, etc.) and the system then proposes suitable movies (Ho, Menezes, and Tagmouti, 2006).

Movie recommender systems have been proposed using various methods and approaches. In the area of design and implementation of a recommender system, there are currently 4 approaches (Adomavicius and Tuzhilin, 2005; Burke, 2007; Falk, 2019; Lu, Wu, Mao, Wang, and Zhang, 2015; Ricci, Rokach, and Shapira, 2011):

- Content-based recommender systems: these systems search for similar product information, services, and other types of content based on their metadata, which were viewed or rated by the user. In these systems, user feedback and rating are the key factors in creating a suitable recommendation for similar products.
- Collaborative filtering recommender systems: these systems create groups of users having similar behavior or preferences to recommend products, services, information, and other types of content, which were positively rated by the group of users to which the user belongs.
- Knowledge-based recommender systems: these systems create a user profile to identify relationships between user preferences and products, information, services, and other types of content.
- Hybrid recommender systems: these systems combine various techniques and algorithms to improve the final recommendation for a given user.

Before describing various approaches in recommender systems, let us mention several principal issues that might arise in various situations in their implementation and subsequent operation (Aggarwal, 2016; Falk, 2019; Khusro, Ali, and Ullah, 2016).

The first is a cold-start problem, which denotes a state when the system cannot recommend any goods corresponding to a new user's preferences due to lack of information (so-called cold visitor), or the customer has such specific preferences that no recommendation can be created based on the behavior of other users (it concerns a so-called gray sheep). In addition, there might be new products added to the system (the so-called cold product). As the product does not have any relationship to other products yet, it is not displayed in nonpersonalized or personalized recommendations. The gray-sheep problem usually arises when the organization focuses on products where the user rating is very subjective, e.g., sale of artworks or paintings. A user that loves Leonardo da Vinci's paintings will not necessarily love his statues, but some users do. A cold visitor concerns a situation when the system does not have any information about the user that relates to their user preferences. The difference between these terms is obvious – cold start, in general, describes a state of the system when a new customer/product appears – there is no possibility of recommending relevant content. The other terms describe more particular cases that cause cold start, either by missing relations between users and products, specific user behavior or lack of information about the users.

Another challenge of recommender systems to be solved is sparsity. The availability of a large quantity of data and users' unwillingness to rate items results in a scattered user-item matrix and decreases recommendation accuracy. In collaborative filtering recommender systems, such a sparse rating hampers the calculation of item prediction.

Another issue is scalability. Recommender systems can have difficulty with processing vast data. An example can be a matrix containing 30 million users and 50 million items.

The last issue, which we mention here, is serendipity. It concerns a situation when an unexpected item is recommended to a user. The user is thus surprised, as the item does not correspond to the preferences. Serendipitous methods then focus on finding such recommendations.

Content-based filtering systems (CBFs) basically depend on two data types: a) description and structure of their attribute and b) user profile generated from their feedback on various items. The advantage of the system is its ability to solve the cold-start problem with respect to new users. The serendipity of a system is relatively low, as recommendations are created based on items already rated by the user. System quality and accuracy mostly depend on the ability of extraction and processing of item content to calculate the similarity to other items. It is also given by the ability to create a user profile based on explicit and implicit feedback. In the area of movie recommender systems, a mashup system was published (Elmisery and Botvich, 2011) based on agent-based middleware, which then uses more data sources to refine the recommended movie prediction. Another system is a recommender system designed for Android devices (mostly smartphones and tablets). Its contribution is good movie categorization and explicit ratings by the user (Barragáns-Martínez, Costa-Montenegro, and Juncal-Martínez, 2015). Other systems and their characteristics are described in the literature review (Véras, Prota, Bispo, Prudêncio, and Ferraz, 2015).

Collaborative filtering systems (CFs) are divided into two basic groups: neighborhood-based collaborative filtering and model-based filtering. The methods for the calculation in the former group are inspired by the nearest neighbor classifications and regression methods. The latter group uses latent factor models to predict item evaluation. Disadvantages of the collaborative filtering system are the cold-start problem, sparsity and scalability. The advantage of the system is the fact that it does not need to understand the products as well as users. Its functionality uses a combination of a user, product, and rating given by the user. System quality and accuracy depend on the ability to decrease the influence of sparsity, e.g., using dimensionality reduction or graph-based models. It is also given by the ability to find latent factors. There is also a movie recommender system Film-Conseil, whose part is a machine learning algorithm to detect whether the user is a good or bad advisor. This algorithm substitutes the missing functionality of explicit movie ratings (Perny and Zucker, 2001). Another movie recommender system uses machine learning techniques called inductive learning. This technique enables decreased sparsity and scalability in the performed experiments (Li and Yamada, 2004). Another published system uses an item-based approach and describes the possibilities of this approach to decrease sparsity and cold-start problems. However, the system was verified on a very small data sample (Ponnam, Punyasamudram, Nallagulla, and Yellamati, 2016). Another movie recommender system uses a cuckoo search, using cluster and optimization-based techniques to improve movie prediction accuracy. The proposed system was verified on the MovieLens dataset and achieved better results compared with other systems under the metrics MAE, RMSE, SD and t-value (Katarya and Verma, 2017). Other systems and their characteristics are described in the literature review (Véras et al., 2015).

Knowledge-based systems, in general, create domains where items are highly specialized and adapted to user needs. It is thus more difficult to determine user preferences based only on item ratings. It is important to give the user more control over the recommender process and emphasize the ability of higher system interactivity. Knowledge-based systems are mostly based on processing users' requirements, and the recommendation uses a small quantity of users' historical data. However, there have been systems that process more user data with a higher degree of personalization. Their advantage is an effective solution to the cold-start problem. A disadvantage lies in a potential problem when acquiring knowledge due to the need to define recommender rules in an explicit, usually expert, way. The movie recommender systems also contain RecomMetz, which is a knowledge-based context-aware recommender system for recommending suitable movies at the cinema at the time of their projection. The system works with three parameters: location, time, and crowd. The system, based on experimental verification, shows quite high values of metrics precision, recall and F-measure (Colombo-Mendoza, Valencia-García, Rodríguez-González, Alor-Hernández, and Samper-Zapater, 2015). Another system is a social knowledge-based recommender system combining a knowledge-based approach with social networks. The main part of the system is the RefSim matrix, which can obtain movie similarity based on information on favorite genres, actors, and directors. The system also performs experimental verification on various types of users in various social network profiles – conservative, moderate, and liberal (Carrer-Neto, Hernández-Alcaraz, Valencia-García, and García-Sánchez, 2012).

The above-described systems (content-based filtering, collaborative filtering, knowledge-based) have various advantages, and many of them have been successfully implemented and published, which is also provided in the lines above. Other systems are described in the literature review (Véras et al., 2015). The systems also have disadvantages and weaknesses. For instance, knowledge-based systems can generally solve the cold-start problem more effectively than content-based filtering or collaborative filtering systems. However, they fall behind in persistent personalization based on historical user data compared with content-based filtering and collaborative filtering systems. Thus, hybrid systems were developed so that they could take advantage of individual approaches.

Hybrid systems combine basic approaches (content-based filtering, collaborative filtering, knowledge-based) and thus can increase the performance and refine the recommendation of items. The main motivation for creating hybrid systems is the fact that current approaches have weaknesses. Hybrid systems focus on their removal while increasing system effectivity. Hybrid systems are divided into three basic groups: ensemble systems, monolithic systems, and mixed systems. In the area of movie recommender systems, there have been several publications. In (Rombouts and Verhoef, 2002), the authors presented a simple hybrid system combining the content-based filtering approach with the collaborative filtering approach, and the user sees explanations for the recommended items. The system uses the advantages of both approaches and has been verified on data from the Netflix database and IMDb. The E-MRS system combines the content-based filtering approach and collaborative filtering together with recommending movies based on user emotions. (Ho et al., 2006). The CinemaScreen Recommender Agent also combines the content-based filtering approach with the collaborative filtering approach. The system has been verified on the MovieLens dataset and shows the prediction improvement when using the hybrid approach in comparison with the traditional collaborative filtering approach or the content-based one (Salter and Antonopoulos, 2006). Another system combining the content-based filtering approach and collaborative filtering is MoviExplain. This system contains an explanation of the recommendation to the user. The explanations are based on rating data together with content data. The system was verified on the MovieLens dataset and compared with similar systems (Symeonidis et al., 2009). Another movie recommender system is MOVREC combining the content-based filtering approach and the collaborative filtering approach. In this system, the user can select values of various attributes (genre, actor, director, year, rating), and the system only proposes a list of recommended movies based on the cumulative weight of different attributes and uses the k-means algorithm (Kumar, Yadav, Singh, and Gupta, 2015). Other hybrid systems are described in the literature review (Véras et al., 2015). The preceding discussion and literature (Aggarwal, 2016; Falk, 2019) shows that hybrid systems can effectively solve certain problems of traditional approaches. Therefore, the focus of our article is on the proposal and development of a hybrid recommender system.

Part of our proposed system is a fuzzy expert system to evaluate movie importance for the final list of recommended movies. The following section describes the possibilities of using fuzzy logic in recommender systems.

2.2. Fuzzy logic in recommender systems

Fuzzy logic in relation to the recommender systems allows for the elaboration of uncertainty, which is based on subjective rating, vagueness, and inaccuracies in the data, e.g., when evaluating user behavior or creating user profiles (Lu et al., 2015), (Wu, Zhang, and Lu, 2015), (Zenebe and Norcio, 2009).

Fuzzy logic was successfully used in recommender systems, where user preferences and object properties are represented by fuzzy sets (Yager, 2003) or in the design of the most suitable items recommended based on incomplete or certain information (Wu et al., 2015).

Recommender systems based on fuzzy logic have been developed, e.g., in the field of e-commerce (Martinez, Barranco, Perez, and Espinilla, 2008), to recommend suitable products, such as books, music, and movies, or to design suitable consumer products, such as mobile phones, tablets, and computers (Cheng and Shen, 2016; Ojokoh, Omisore, Samuel, and Ogunniyi, 2012; Oramas, Ostuni, Noia, Serra, and Sciascio, 2017; Subramaniyaswamy, Logesh, Chandrashekhar, Challa, and Vijayakumar, 2017; Xiao and Benbasat, 2014).

A study (Parra and Amatriain, 2011) based on a comparison of various approaches in the area of user approaches determined that the best approaches in the area of user approaches are based on direct feedback from the users.

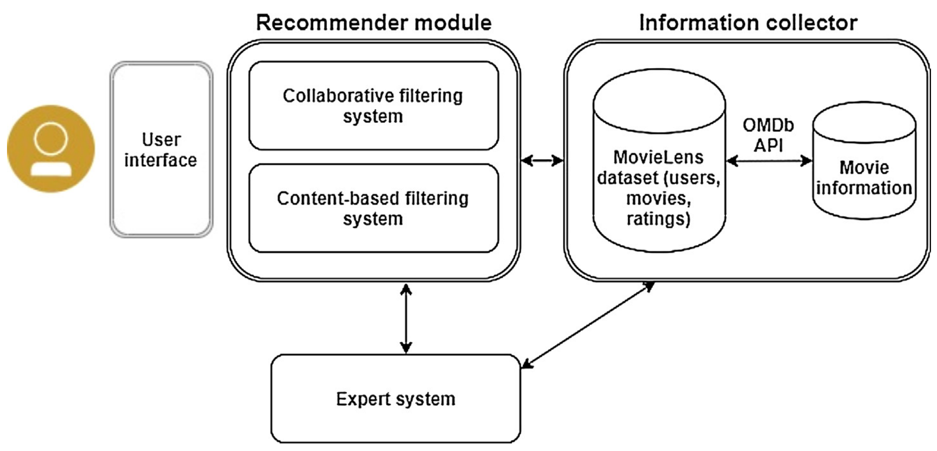
Recommender module
Information collector
User interface
Collaborative filtering system
Content-based filtering system
MovieLens dataset (users, movies, ratings)
OMDb API
Movie information
Expert system

Fig. 1. Architecture of the proposed recommender system Predictory.

Recommender systems based on fuzzy logic, which use direct information (user rating or feedback), have been published and developed for various problem domains (Herrera-Viedma, Porcel, Lopez-Herrera, and Alonso, 2008; Wu et al., 2015; Zhang et al., 2013).

The discussion above shows that using fuzzy logic in recommender systems is suitable when incomplete or vague information occurs or if processing vague terms is needed.

In this paper, we propose a monolithic hybrid recommender system composed of a collaborative filtering system, a content-based filtering system, and a fuzzy expert system. The proposed system combines current traditional approaches as well as an expert system to support decision making with respect to a prefinal recommendation of items for a user.

3. Recommender system

This section describes our proposed recommender system. Our system is a monolithic hybrid system connected with an expert system for the final ordering of recommendations, in this case, movies. The system takes advantage of collaborative filtering and content-based systems to construct the recommender module. Our proposed recommender system is fully implemented in the form of a web system called Predictory. The architecture of the proposed system is depicted in Fig. 1.

The architecture of the described system consists of several main modules:

- User interface
- Recommender module
  - Collaborative filtering system
  - Content-based filtering system
- Expert system
- Information collector

The methodology of the proposed system is depicted in Fig. 2.

3.1. Information collector

To suitably display a recommendation based on the collaborative filtering algorithm, other users' movie ratings are necessary. That was the reason to focus on finding a suitable database of movie ratings from various users. For the purposes of the proposal and implementation of our proposed recommender system, we selected the MovieLens dataset (Harper and Konstan, 2016; MovieLens, 2019), which is a database of personalized ratings of various movies from a large number of users. This database was developed by a research lab at the University of Minnesota. The dataset used for the development of the proposed recommender system contains 100,836 ratings made by 610 users on 9724 movies. This dataset contains users' personalized ratings so that each rating is assigned to a particular user. This enables us to determine which movie genres the user rated, which genres the user rates most often, and the average movie rating of the user across various genres.

This dataset is then extended with another 40 users and their item ratings, which consists of users who took part in the testing of our proposed recommender system (23 who participated in previous testing stages – 14 out of whom compared the system with other open-source solutions, see Section 4). The users are aged 18 – 35 with various preferences of favorite movies and genres.

To acquire recommendations in our system, we also use a whole set of movie ratings available in the MovieLens dataset (100,836 ratings) extended with the ratings of the 40 above-mentioned users. These data are used within the collaborative filtering system described in Section 3.2.1, which is part of our proposed hybrid system. Its outputs provide grounds for the next steps in our methodology.

The resulting MovieLens dataset contains movie ratings, which can be visualized using a user-item matrix, as provided in Table 1.

The table shows that in the selected sample of users and movies, only some users rated some movies. This fact corresponds to the current state of movie ratings across various platforms and servers for movie ratings. For instance, User3 rated only 2 out of 5 selected movies, whereas, User1, User2, and User4 rated 4 out of 5 selected movies. Only two of the selected movies were rated by the majority of the users (6 out of 7 users rated the movies). The rating has a range of intervals $\langle 0.5, 5\rangle$, number of stars, and it is possible to rate by whole stars or half stars. Therefore, the rating can be 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, or 5, where 0.5 represents the worst rating and 5 represents the best rating. The experimental data sample does not have a complete rating of all movies by all users, which results in the sparsity problem.

Although the MovieLens dataset contains a large number of user ratings of many movies, it only contains basic information about the movies (name, category, movie ID, IMDb movie ID). This is the reason more information is added to the database using a service called OMDb API (OMDb API, 2019). This service provides a RESTful API interface enabling the acquisition of additional information about a movie based on the IMDb movie ID. The API provides the following information:

- Name
- Year and date of release
- Genres

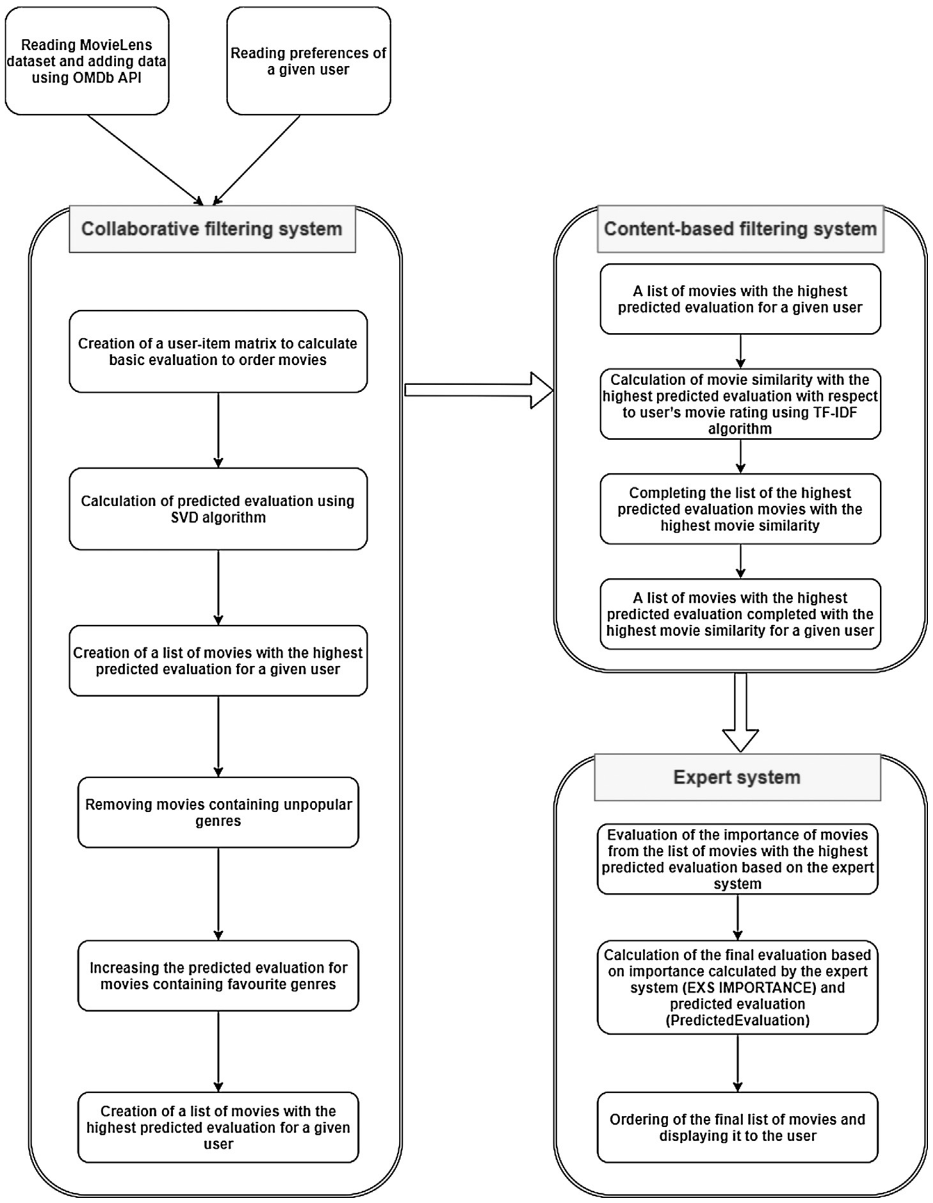
Reading MovieLens dataset and adding data using OMDb API
Reading preferences of a given user
Collaborative filtering system
Creation of a user-item matrix to calculate basic evaluation to order movies
Calculation of predicted evaluation using SVD algorithm
Creation of a list of movies with the highest predicted evaluation for a given user
Removing movies containing unpopular genres
Increasing the predicted evaluation for movies containing favourite genres
Creation of a list of movies with the highest predicted evaluation for a given user
Content-based filtering system
A list of movies with the highest predicted evaluation for a given user
Calculation of movie similarity with the highest predicted evaluation with respect to user's movie rating using TF-IDF algorithm
Completing the list of the highest predicted evaluation movies with the highest movie similarity
A list of movies with the highest predicted evaluation completed with the highest movie similarity for a given user
Expert system
Evaluation of the importance of movies from the list of movies with the highest predicted evaluation based on the expert system
Calculation of the final evaluation based on importance calculated by the expert system (EXS IMPORTANCE) and predicted evaluation (PredictedEvaluation)
Ordering of the final list of movies and displaying it to the user

Fig. 2. Methodology of the proposed recommender system.

Table 1
User-item matrix containing ratings of selected movies by selected users.

	User1	User2	User3	User4	User5	User6	User7
Toy Story	3.5	2.5	4	5	3.5	3	
Jumanji	3	2.5		5	4	3	2.5
The Martian	4.5	5	4				
Kingsman: The Golden Circle	3.5			3			
Wimbledon		4.5		3	2.5	3	1.5

- Director
- Actors
- Languages
- Movie description

The method for reading the movie data is depicted in Fig. 3.

Having read the data from the MoviesLens dataset and added information from the OMDb API, the data about the users, their ratings, and movies were stored in the relational database MariaDB.

3.2. Recommender module

The recommender module is a recommender system of a monolithic hybrid type. It contains 2 subsystems – a collaborative filtering system and a content-based filtering system. The collaborative filtering system serves to read suitable movies ranked by other users together with movie ratings of the given user. The content-based filtering system then serves to read similar movies with respect to the best-rated movies by the user. The output of the two systems is then processed using an expert system for the

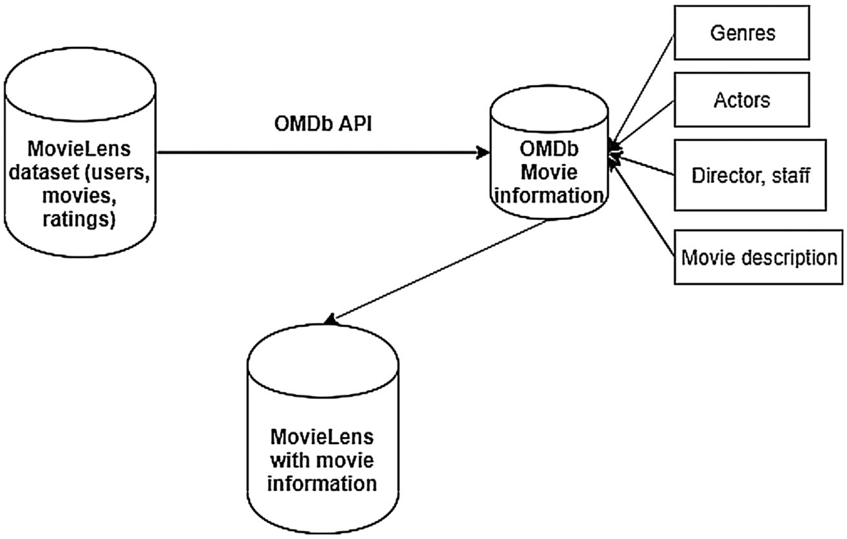
MovieLens dataset (users, movies, ratings)
OMDb API
OMDb Movie information
Genres
Actors
Director, staff
Movie description
MovieLens with movie information

Fig. 3. Method for reading movie information from the OMDb database.

final ranking of the most suitable movies to display to the user. Both systems are described in more detail in the following subsections.

3.2.1. *Collaborative filtering system*

The collaborative filtering system is one of the main parts of the whole system. It attempts to create as good of a user-item matrix as possible; in our case, it consists of two basic facts:

- User – user movie rating, numerical movie ranking (interval range ⟨0.5,5⟩) by a single user
- Item – rated movie

This user-item matrix serves to calculate the rating, which is used to rank the most suitable movies for the user based on their movie ratings and ratings of other users. Currently, there are two basic groups of collaborative filtering systems (Falk, 2019):

- Neighborhood-based (memory-based) collaborative filtering
- Model-based collaborative filtering

In the case of a neighborhood-based system, the main objective is to calculate the expected user rating based on the similarity between users and items. There are two approaches in this group: user-based approaches and item-based approaches. The user-based approach aims to find a user with similar preferences and then recommend the most relevant item that the given users have not yet seen. The item-based approach aims at finding similar items to those that the user already rated and thus at recommending others.

In the case of the model-based system, the main objective is to find latent factors (hidden genres) in the data. This can be achieved using matrix decomposition, e.g., using the method of single-value decomposition (SVD).

The recommendation takes place in two ways. The first possibility is to calculate all ratings for the given user, their ordering and the subsequent recommendation of the top-N products with the highest rating.

The second possibility is a combination with neighborhood-based filtering. However, instead of using original data, it uses found latent factors and then searches for similarity to them.

To choose an appropriate group of a collaborative filtering system and the final algorithm, we performed a comparison of individual algorithms in both groups. A relevant indicator is a mean prediction error according to the RMSE algorithm (the lower the error is, the more accurately the algorithm predicts missing values in the user-item matrix).

Table 2
Test results for the user-based approach.

Run	Error
1	3.176
2	3.185
3	3.177
4	3.182
5	3.176
6	3.177
7	3.190
8	3.185
9	3.173
10	3.172

Table 3
Test results for the item-based approach.

Run	Error
1	3.397
2	3.397
3	3.403
4	3.399
5	3.395
6	3.395
7	3.397
8	3.389
9	3.392
10	3.387

The mean prediction error for individual algorithms was calculated as a mean of errors of individual runs in the test data using a given algorithm. Each algorithm had 10 runs on the MovieLens dataset containing 100,836 ratings by 610 users on 9724 movies. These data were randomly divided into 75% for training and 25% for verification. The results of individual runs for individual algorithms are provided in Tables 2–4.

Table 5 contains the calculated mean prediction error according to the RMSE algorithm for all tested algorithms.

Table 5 clearly shows that the lowest mean prediction error according to the RMSE algorithm was scored by SVD – single-value decomposition. This is why the collaborative filtering system uses

Table 4
Test results for single-value decomposition.

Run	Error
1	3.059
2	3.066
3	3.050
4	3.057
5	3.059
6	3.062
7	3.070
8	3.059
9	3.066
10	3.050

Table 5
Mean errors of the algorithms in the test.

Algorithm	Mean prediction error according to RMSE
User-based approach	3.179
Item-based approach	3.395
Single Value Decomposition (SVD)	3.060

Table 6
Part of the user-item matrix containing only movie ratings by the users.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
U1						3.5	5.0	5.0	3.5	1.5
U2									3.5	1.5
U3								4.5		
U4	5.0							4.5		4.5
U5		5.0								
U6						4.5				
U7		3.0	3.0							
U8		5.0	5.0							
U9		4.0	3.5							
U10		5.0	5.0							

Table 7
Part of user-item matrix containing movie rating of the users added with zero values.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
U1	0	0	0	0	0	3.5	5.0	5.0	3.5	1.5
U2	0	0	0	0	0	0	0	0	3.5	1.5
U3	0	0	0	0	0	0	0	4.5	0	0
U4	5.0	0	0	0	0	0	0	4.5	0	4.5
U5	0	5.0	0	0	0	0	0	0	0	0
U6	0	0	0	0	0	4.5	0	0	0	0
U7	0	3.0	3.0	0	0	0	0	0	0	0
U8	0	5.0	5.0	0	0	0	0	0	0	0
U9	0	4.0	3.5	0	0	0	0	0	0	0
U10	0	5.0	5.0	0	0	0	0	0	0	0

the SVD algorithm from the group model-based collaborative filtering.

SVD is a method of decomposing matrix M into individual components for the purposes of simplification of further calculations (Falk, 2019). The outputs of SVD are three matrices – U, Σ and V^T, where

- M - a matrix we want to decompose, in our case, the rating matrix of all ratings of movies by users
- U - user feature matrix; user is a user who evaluates movies
- Σ – the weights diagonal matrix provides information about how much we should reduce the dimensions
- V^T - item feature matrix; in our case, item is a movie and ^{T} represents a specific rated movie

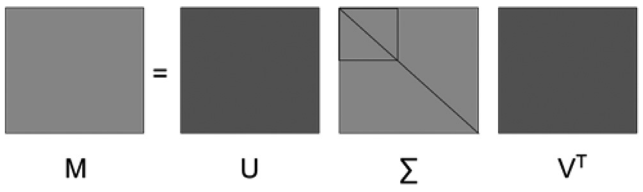

Fig. 4. Process of decomposing matrix M (source: (Falk, 2019)).

When using the SVD algorithm, Σ will always be a diagonal matrix (Falk, 2019).

Rating acquisition then takes place using a scalar product U and Σ V^T matrices on a given position. The process of decomposing matrix M is depicted in Fig. 4.

The principle of the algorithm function is depicted in Fig. 5.

A problem with the SVD algorithm is that it cannot work with a matrix with missing values (it considers these values as 0); therefore, we add the values into the matrix as described further down in the text.

The principle of the collaborative filtering system function is schematically depicted in Fig. 6.

The first step of the collaborative filtering system is to read unique user IDs, unique movie IDs, and user movie ratings from the MovieLens dataset and to calculate the number of users and movies to create a matrix. Then, a user-item matrix is created containing users in rows and movies in columns. The values are ratings of individual movies. This process is described in Algorithm 1.

The next step consists of using the SVD algorithm to calculate matrix U, Σ, and V^T and a matrix containing the predicted movie rating, which predicts the movie ratings of the users.

Table 6 shows a part of the user-item matrix, which contains (for the purposes of SVD) 650 rows (MovieLens users plus our experimental users) and 9724 columns (number of rated movies in the MovieLens dataset). For the demonstration purposes of the algorithm work, we selected a part of the user-item matrix containing 10 rows (users) and 10 columns (movies). The user-item matrix containing only movie ratings of the users is shown in Table 6.

For clarity, the table contains aliases for users and movies:

U1 =	user ID 1500
U2 =	user ID 1504
U3 =	user ID 1507
U4 =	user ID 1510
U5 =	user ID 1517
U6 =	user ID 1532
U7 =	user ID 1536
U8 =	user ID 1539
U9 =	user ID 1545
U10 =	user ID 1547
M1 =	movie The Matrix
M2 =	movie The Lord of the Rings: The Fellowship of the Ring
M3 =	movie The Lord of the Rings: The Return of the King
M4 =	movie Inception
M5 =	movie The Dark Knight Rises
M6 =	movie Iron Man 3
M7 =	movie Guardians of the Galaxy
M8 =	movie Black Panther
M9 =	movie Guardians of the Galaxy Vol. 2
M10 =	movie Tomb Raider

Table 6 clearly shows that the user-item matrix contains only several ratings for each movie. Movies M4 and M5 were not rated by any user in this part of the user-item matrix. As most ratings are missing in this user-item matrix, it is necessary to add the data automatically so that the SVD algorithm can run properly. Having no information about the missing values, the value is set to zero. The user-item matrix, added with zero values, is shown in Table 7.

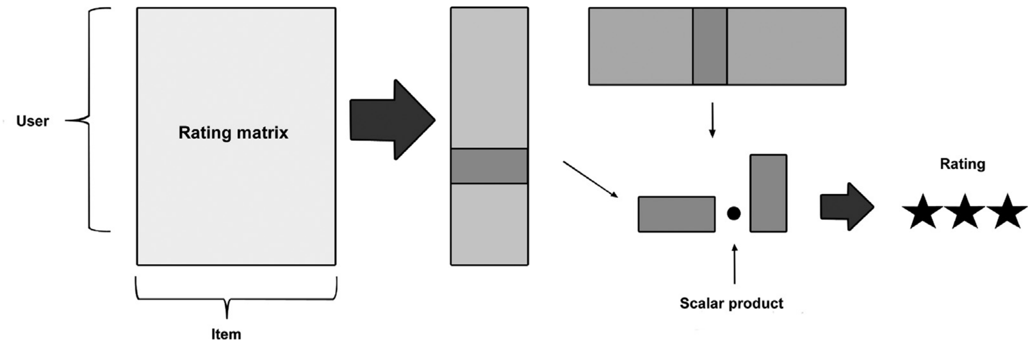

Fig. 5. Principle of the SVD algorithm function in the collaborative filtering system.

Algorithm 1 Creation of the user-item matrix.

```
Input:
M represents a set of all movies in the MovieLens dataset {$M_1$, $M_2$, $M_3$,..., $M_n$},
U represents a set of all users rating movies – users in the MovieLens dataset + other users who took part in testing the system {$U_1$, $U_2$, $U_3$..., $U_n$},
R represents a set of all ratings by users from set U on movies from set M
Output:
User movie rating matrix (UMR Matrix) - User-item matrix with movie ratings
  MoviesNum = count(M) //number of movies
  UsersNum = count(U) //number of users
  User movie rating matrix = matrix UsersNum * MoviesNum
  //create matrix
  foreach($U_i$) in Users do //For all users
  {
    foreach($M_j$) in Movies do //For all movies
    {
      if(Rating(i,j) != null) UMR matrix(i,j) = Rating(i,j);
      //Read user and movie rating – user rating for given movie, if exists,
      insert rating into matrix
      else UMR matrix(i,j) = 0;
    //else store null rating into given matrix cell to decrease the sparsity problem
    }
  }
```

Now the system calculates the predicted evaluation based on Algorithm 2:

Algorithm 2 Calculation of the predicted evaluation.

```
1. Load the user-item matrix of all users, all movies and their ratings
2. Add zero values to missing ratings in the user-item matrix
3. Set dimension $k = 20$ to reduce matrices U, $\Sigma$, and $V^T$
4. Calculate matrices U, $\Sigma$, and $V^T$ using SVD
5. Reduce matrices U, $\Sigma$, and $V^T$ and calculate matrices containing predicted
evaluation for all users
```

A matrix containing the predicted evaluation for a selected part of the user-item matrix is depicted in Table 8.

The following steps in our proposed approach are demonstrated on an example of an existing user with ID 1500. This user rated 20 movies and then selected 3 favorite and 3 unpopular genres of movies. His ratings are depicted in Table 9.

The bold type in the table marks the ratings that are part of the user-item matrix in Table 6. Table 10 shows selected favorite and unpopular genres of the user.

Now we select only the row of our current user with ID 1500 from the predicted evaluation matrix. As we want to propose only suitable movies that he has not rated yet, we remove all movies that he has already rated. The resulting list of predicted evaluations is ordered from the highest rating in descending order. The result is thus a list containing 9704 predicted evaluations for our user (9724 movies decreased by 20 already rated movies). We select only 25 with the highest predicted evaluation, as shown in Table 11.

The bold type in the table marks movies that are in the top 25 of movies with the highest predicted evaluation, yet they belong to the genres that the user marked as unpopular. The system works with a hypothesis – “do not recommend movies from my unpopular genres”. Therefore, those movies are removed from the list, and the list is then completed with other relevant movies ranked 26th and lower in descending order. The table also depicts bold predicted evaluations that are displayed to users with ID 1500 in Table 6. A modified list is shown in Table 12.

The next step consists of using the genres that the user marked as favorite. Movies of favorite genres are preferred by the user, so their evaluation is increased. At this point, the system works with the hypothesis – “recommend primarily movies from my favorite genres”. The evaluations of movies of favorite genres are calculated based on the following:

$$FavouriteGenrePredictedEvaluation = PredictedEvaluation \times 2$$

A modified list with updated predicted evaluation for favorite genres is shown in Table 13.

The updated FavoriteGenrePredictedEvaluation is performed for all movies with predicted evaluations so that movies with formerly lower predicted evaluations could have higher evaluations after the update, which results in their inclusion in the list of movies with the highest predicted evaluations substituting for high-prediction movies, but they do not belong to the user’s favorite genres. In our case, this situation occurred in the 4 movies marked in bold (Gladiator, The Martian, Edge of Tomorrow, Star Wars: The Force Awakens). These movies contain the users’ favorite genres while having quite a high original predicted rating, which means that they are relatively highly rated by other users.

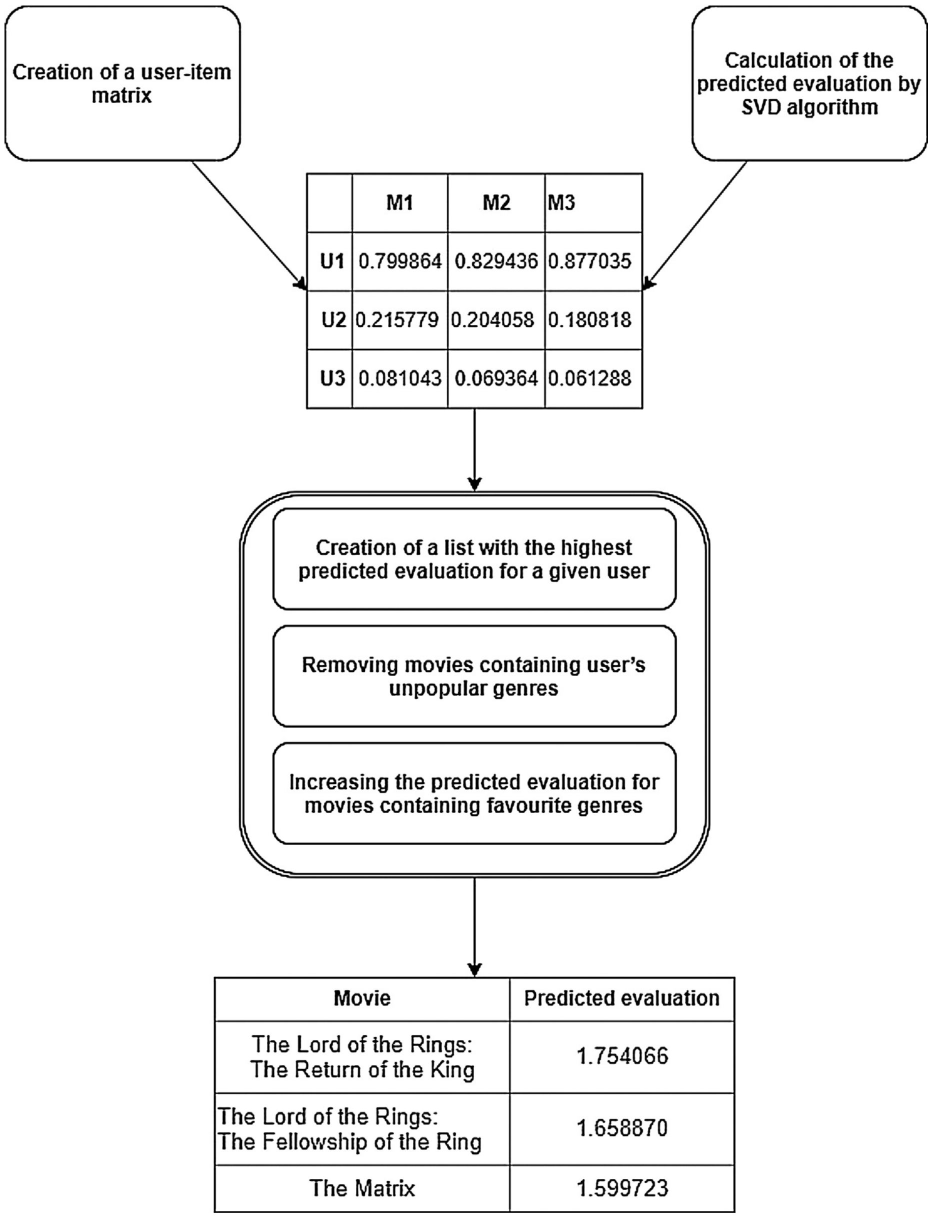
Creation of a user-item matrix
Calculation of the predicted evaluation by SVD algorithm
M1
M2
M3
U1
0.799864
0.829436
0.877035
U2
0.215779
0.204058
0.180818
U3
0.081043
0.069364
0.061288
Creation of a list with the highest predicted evaluation for a given user
Removing movies containing user's unpopular genres
Increasing the predicted evaluation for movies containing favourite genres
Movie
Predicted evaluation
The Lord of the Rings: The Return of the King
1.754066
The Lord of the Rings: The Fellowship of the Ring
1.658870
The Matrix
1.599723

Fig. 6. Principle of collaborative filtering system functioning.

Table 8
Part of the user-item matrix containing predicted evaluation based on the SVD algorithm calculation.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
U1	0.79	0.82	0.87	0.86	0.56	0.28	0.48	0.17	0.34	0.06
U2	0.21	0.20	0.18	0.35	0.29	0.19	0.32	0.13	0.23	0.04
U3	0.08	0.06	0.06	0.09	0.06	0.04	0.07	0.04	0.06	0.02
U4	0.49	0.37	0.34	0.29	0.15	0.10	0.13	0.09	0.15	0.04
U5	0.73	0.74	0.73	0.51	0.25	0.12	0.19	0.08	0.12	0.04
U6	0.31	0.23	0.22	0.37	0.23	0.09	0.19	0.05	0.14	0.03
U7	0.52	0.72	0.74	0.55	0.38	0.21	0.37	0.09	0.16	0.03
U8	0.85	1.37	1.40	0.61	0.38	0.27	0.40	0.11	0.19	0.05
U9	0.56	0.78	0.82	0.55	0.38	0.20	0.36	0.09	0.16	0.03
U10	0.93	1.16	1.19	0.55	0.35	0.26	0.31	0.11	0.20	0.05

Table 9
Movie ratings by the user with ID 1500.

Name of movie	Rating
Iron Man 3	**3.5**
Iron Man 2	4.0
Black Panther	**5.0**
Pacific Rim: Uprising	4.0
Jumanji: Welcome to the Jungle	5.0
Guardians of the Galaxy Vol. 2	**3.5**
Avengers: Infinity War	5.0
Jurassic World: Fallen Kingdom	3.5
Guardians of the Galaxy	**5.0**
Jurassic Park	5.0
The Avengers	5.0
The Shawshank Redemption	5.0
Mission: Impossible – Fallout	4.0
Forrest Gump	2.5
Tomb Raider	**1.5**
Fifty Shades of Grey	1.0
The Dark Knight	4.5
BlacKkKlansman	4.0
The Lord of the Rings: The Two Towers	5.0
Spider-Man: Homecoming	5

Table 10
Selection of favorite and unpopular genres by the user with ID 1500.

Favourite genres	Unpopular genres
Adventure	Romance
Sci-fi	Crime
Fantasy	Thriller

Table 11
List of movies with the highest predicted evaluation for users with ID 1500.

Name of movie	Predicted evaluation
The Lord of the Rings: The Return of the King	**0.877032**
Inception	**0.866288**
The Lord of the Rings: The Fellowship of the Ring	**0.829434**
The Matrix	**0.799861**
Pulp Fiction	0.681752
The Silence of the Lambs	0.681753
Fight Club	0.627434
Up	0.590796
The Dark Knight Rises	0.563551
WALL•E	0.539541
Iron Man	0.539085
Interstellar	0.513695
Schindler's List	0.509111
Star Wars: Episode IV - A New Hope	0.507423
Braveheart	0.491871
The Lion King	0.462482
Raiders of the Lost Ark	0.461376
Toy Story	0.441814
Star Wars: Episode V - The Empire Strikes Back	0.426589
Deadpool	0.422765
Memento	0.420562
Inglorious Basterds	0.419951
The Usual Suspects	0.407013
Batman Begins	0.395846
Avatar	0.394044

Table 12
Modified list of movies with the highest predicted evaluation for users with ID 1500.

Name of movie	Predicted evaluation
The Lord of the Rings: The Return of the King	0.877033
The Lord of the Rings: The Fellowship of the Ring	0.829434
The Matrix	0.799861
Fight Club	0.627434
Up	0.590796
WALL·E	0.539541
Iron Man	0.539085
Interstellar	0.513695
Schindler's List	0.509111
Star Wars: Episode IV - A New Hope	0.507423
Braveheart	0.491871
The Lion King	0.462482
Raiders of the Lost Ark	0.461376
Toy Story	0.441814
Star Wars: Episode V - The Empire Strikes Back	0.426589
Deadpool	0.422765
Inglorious Basterds	0.419951
Batman Begins	0.395846
Avatar	0.394044
Apollo 13	0.386608
Pirates of the Caribbean: The Curse of the Black Pearl	0.384754
The Incredibles	0.384159
Django Unchained	0.371304
Shrek	0.367649
Star Trek	0.362435

Table 13
A modified list of movies with the highest predicted evaluation and updated evaluation for movies of favorite genres.

Name of movie	Predicted evaluation
The Lord of the Rings: The Return of the King	1.754066
The Lord of the Rings: The Fellowship of the Ring	1.658870
The Matrix	1.599723
Up	1.181593
WALL•E	1.079083
Iron Man	1.078170
Interstellar	1.027390
Star Wars: Episode IV - A New Hope	1.014846
The Lion King	0.924965
Raiders of the Lost Ark	0.922753
Toy Story	0.883628
Star Wars: Episode V - The Empire Strikes Back	0.853178
Deadpool	0.845530
Inglorious Basterds	0.839903
Batman Begins	0.791693
Avatar	0.788088
Apollo 13	0.773216
Pirates of the Caribbean: The Curse of the Black Pearl	0.769508
The Incredibles	0.768319
Shrek	0.735298
Star Trek	0.724870
Gladiator	0.722487
The Martian	0.707388
Edge of Tomorrow	0.689725
Star Wars: The Force Awakens	0.681560

3.2.2. Content-based filtering system

The second subsystem of the recommender module is a content-based filtering system. This system serves to calculate movie similarity to the highest predicted evaluation with respect to movies that the user has rated. The input into this system is the final list of 25 movies with the highest predicted evaluation, which is the output from the collaborative filtering system.

A content-based filtering system generally consists of several components: a) preprocessing and feature extraction, b) content-based learning of user profiles, and c) filtering and recommendation (Aggarwal, 2016; Falk, 2019). Within the preprocessing and feature extraction component, the most commonly used algorithms are TF-IDF and LDA (Falk, 2019). We chose the TF-IDF algorithm for its easy implementation and lower requirements on the system. Despite its simplicity, the algorithm mostly provides comparable results with the LDA algorithm (if n-grams are used). However, unlike in LDA, adding a new product requires repeating the entire training process, whereas, in LDA, the created model can be used repeatedly. Within the content-based learning of user profiles and the area of the nearest neighbor classification, we selected the Cosine similarity function as it represents one of the most used similarity functions. If we worked with structured data, it would be suitable to use other similarity/distance functions, e.g., Euclidean distance or Manhattan distance (Aggarwal, 2016; Pazzani and Billsus, 2007).

The process of calculating the similarity between a particular movie from the list of 25 movies with the highest predicted evaluation and all movies that the user rated is presented in the following steps:

1. The system reads all user's rated movies
2. The system reads a so-called document for each rated movie (in this case, it is the movie description) and creates a so-called bag-of-words. Then, it removes stop words (words causing unnecessary noise), suffixes of individual words, and creates trigrams (n-grams of 3) – i.e., each document has a field of so-called tokens representing its content
3. The TF-IDF algorithm selects the token characterizing the whole model (so-called features) and creates fields containing data in a form (document ID, token ID)
4. Using Cosine similarity, the system calculates the similarity between a particular movie from the list of 25 movies with the highest predicted evaluation and all of the user's rated movies; then, it orders the similarity of the rated movies from the highest to the lowest
5. The system selects a rated movie with the highest similarity and assigns it to a particular movie from the list of 25 movies as a similarity to the rated movies
6. Steps 1–6 are repeated for all 25 movies with the highest predicted evaluation

The process of calculating the similarity between a particular movie from the list of 25 movies with the highest predicted evaluation and all movies rated by the user is depicted using the following pseudocode in Algorithm 3:

Algorithm 3 Process for calculating the similarity.

```
$I = 25$ (25 movies with the highest predicted evaluation)
For $i = 0$ to I
  Load all rated movies by user u
  $J$ = number of user's rated movies
  MaxSim = 0 //highest similarity between movies
  For $j = 0$ to J
    Load document and create bag-of-words from $M_j$
    Create an array of tokens from document
    Load feature tokens from array of tokens using TF-IDF
    Create an array of document
    Compute similarity between $M_i$ and $M_j$
    If($SimM_iM_j$ > MaxSim)
      MaxSim = $SimM_iM_j$
    End If
  End For
$M_j$Sim = MaxSim
End For
```

The resulting list of 25 movies with the highest predicted evaluation added with the highest similarity to the user's rated movies is shown in Table 14.

3.3. Expert system

The second main module in our proposed recommender system is an expert system that serves for the final ranking of the recommended movies. This final ranking is created based on relevant information that can be read within the system for given movies. It concerns the following information:

- Average rating of the movie
- Total number of movie ratings
- Level of similarity to already rated movies – output from the content-based filtering system

A fuzzy expert system was selected due to its ability to model vague terms using fuzzy sets as well as the ability to simply modify the definitions of linguistic variables. The knowledge base of an expert system composed of IF-THEN rules can also be easily modified and later extended. The modification of the fuzzy expert system uses a software tool called the Linguistic Fuzzy Logic Controller (Habiballa, Novák, Dvořák, and Pavliska, 2003).

Based on this information, the following input linguistic variables of the expert system knowledge base were created:

- INP1 – average rating of the movie, values from interval ⟨0,5⟩
- INP2 – total number of movie ratings, values from interval ⟨0, 350⟩
- INP3 – level of similarity to already rated movies, values from interval ⟨0,1⟩

The output linguistic variable is

- IMPORTANCE – signifies the level of importance of a given movie for the final ranking, values from interval ⟨0,1⟩

The expert system was created in the Linguistic Fuzzy Logic Controller (LFLC) (Habiballa et al., 2003). LFLC enables us to define and fill the base of an expert system. It also contains possibilities for selecting the inference mechanism and the defuzzification method to calculate the value of the output linguistic variable. Examples of IF-THEN rules are provided below:

1 IF (INP1 is low) and (INP2 is few) and (INP3 is low) THEN (IMPORTANCE is very low)
2 IF (INP1 is low) and (INP2 is few) and (INP3 is very high) THEN (IMPORTANCE is low)
3 IF (INP1 is medium) and (INP2 is few) and (INP3 is very high) THEN (IMPORTANCE is medium)
4 IF (INP1 is medium) and (INP2 is much) and (INP3 is very high) THEN (IMPORTANCE is medium)
5 IF (INP1 is high) and (INP2 is much) and (INP3 is very high) THEN (IMPORTANCE is high)
6 IF (INP1 is high) and (INP2 is very much) and (INP3 is very high) THEN (IMPORTANCE is very high)
7 IF (INP1 is high) and (INP2 is very much) and (INP3 is very high) THEN (IMPORTANCE is high)

Table 15 contains an illustrative list of selected IF-THEN rules. Individual columns state the linguistic values of the input linguistic variables and the output linguistic variables. Within the inference and defuzzification, the resulting crisp number is calculated, which represents the final numerical value EXS IMPORTANCE – see Table 16. Testing and tuning of our proposed system also included various inference and defuzzification methods. Having performed the tests, we selected the inference method fuzzy approximation with conjunctions and defuzzification method Modified Center of Gravity. A complete knowledge base of the fuzzy expert system contains a total of 144 IF-THEN rules. The complete list is provided in the appendix (Supplementary Material).

Fig. 7 depicts membership functions for the output linguistic variable IMPORTANCE. The red line marks a linguistic variable *high*; other linguistic variables are *very low, low, medium, and very high.*

The list of movies with the highest predicted evaluation is then added with columns average rating, number of ratings, and EXS IMPORTANCE, which is the value of a linguistic variable IMPORTANCE of the expert system. This list is depicted in Table 16.

Based on the experimental results, we found that if the final ranking of the recommended movies were based only on EXS IMPORTANCE, the value of the predicted evaluation would sometimes be suppressed too much. Therefore, the calculation of the final evaluation was performed based on EXS IMPORTANCE and predicted evaluation according to the following:

$\text{If}(\text{PredictedEvaluation} =< 0)\text{FinalEvaluation}$

Table 14
Resulting list of 25 movies with the highest predicted evaluation added with the highest similarity to the rated movies.

Name of movie	Predicted evaluation	Similarity
The Lord of the Rings: The Return of the King	1.754066	0.067945
The Lord of the Rings: The Fellowship of the Ring	1.658870	0.065668
The Matrix	1.599723	0.010632
Up	1.181593	0
WALL·E	1.079083	0.016983
Iron Man	1.078170	0.127942
Interstellar	1.027390	0.028137
Star Wars: Episode IV - A New Hope	1.014846	0.013116
The Lion King	0.924965	0
Raiders of the Lost Ark	0.922753	0
Toy Story	0.883628	0
Star Wars: Episode V - The Empire Strikes Back	0.853178	0
Deadpool	0.845530	0.011877
Inglorious Basterds	0.839903	0.012373
Batman Begins	0.791693	0.054025
Avatar	0.788088	0.010986
Apollo 13	0.773216	0.011421
Pirates of the Caribbean: The Curse of the Black Pearl	0.769508	0
The Incredibles	0.768319	0
Shrek	0.735298	0
Star Trek	0.724870	0
Gladiator	0.722487	0
The Martian	0.707388	0.010971
Edge of Tomorrow	0.689725	0
Star Wars: The Force Awakens	0.681560	0

Table 15
Selected IF-THEN rules of the expert system.

Rule	INP1	INP2	INP3	IMPORTANCE
1	Low	Few	Low	Very low
4	Low	Few	Very high	Low
52	Medium	Few	Very high	Medium
76	Medium	Much	Very high	Medium
132	High	Much	Very high	High
140	High	Very much	Very high	Very high
144	High	Very much	Very high	High

$$= \text{PredictedEvaluation} \times \text{EXS IMPORTANCE}$$

$$\text{If}(\text{PredictedEvaluation} > 0)\text{FinalEvaluation}$$
$$= \text{PredictedEvaluation} \times (1 + \text{EXS IMPORTANCE})$$

The value of PredictedEvaluation is lower than 0 in cases when the user makes few ratings or it concerns the calculation of PredictedEvaluation in movies that do not correspond to the user's preferences. In such a case, the FinalEvaluation value is 0 or lower, which means that such movies will not usually appear in the final list of recommended movies for a user. In the second branch of the formula, the value of PredictedEvaluation is multiplied by

Table 16
List of 25 movies with the highest predicted evaluation added with columns average rating, number of ratings, and EXS IMPORTANCE.

Name of movie	Predicted evaluation	Similarity	Average rating	Number of ratings	EXS IMPORTANCE
The Lord of the Rings: The Return of the King	1.754066	0.067945	4.085492	193	0.75
The Lord of the Rings: The Fellowship of the Ring	1.658870	0.065668	4.082125	207	0.75
The Matrix	1.599723	0.010632	4.199646	283	0.75
Up	1.181593	0	4.0	106	0.5
WALL•E	1.079083	0.016983	4.057692	104	0.75
Iron Man	1.078170	0.127942	3.843750	96	0.74
Interstellar	1.027390	0.028137	4.019736	76	0.72
Star Wars: Episode IV - A New Hope	1.014846	0.013116	4.235177	253	0.75
The Lion King	0.924965	0	3.963068	176	0.5
Raiders of the Lost Ark	0.922753	0	4.207500	200	0.5
Toy Story	0.883628	0	3.924311	218	0.5
Star Wars: Episode V - The Empire Strikes Back	0.853178	0	4.220657	213	0.5
Deadpool	0.845530	0.011877	3.693548	62	0.51
Inglorious Basterds	0.839903	0.012373	4.136363	88	0.74
Batman Begins	0.791693	0.054025	3.862068	116	0.74
Avatar	0.788088	0.010986	3.586734	98	0.50
Apollo 13	0.773216	0.011421	3.851485	202	0.74
Pirates of the Caribbean: The Curse of the Black Pearl	0.769508	0	3.753333	150	0.5
The Incredibles	0.768319	0	3.846456	127	0.5
Shrek	0.735298	0	3.901129	177	0.5
Star Trek	0.724870	0	3.864406	59	0.49
Gladiator	0.722487	0	3.939306	173	0.5
The Martian	0.707388	0.010971	4.010204	49	0.50
Edge of Tomorrow	0.689725	0	3.977777	45	0.49
Star Wars: The Force Awakens	0.681560	0	3.852272	44	0.49

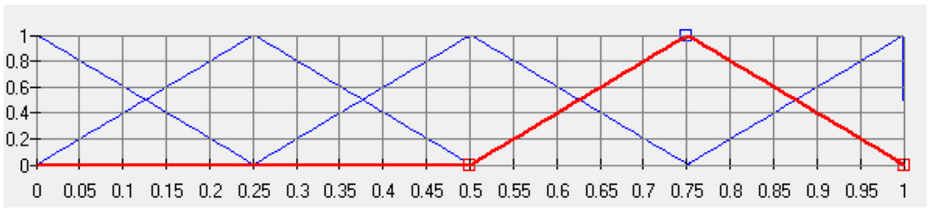

Fig. 7. Membership functions for output linguistic variable IMPORTANCE.

Table 17
Final list of 25 recommended movies.

Name of movie	Predicted Evaluation	EXS IMPORTANCE	Final evaluation
The Lord of the Rings: The Return of the King	1.754066	0.75	3.069
The Lord of the Rings: The Fellowship of the Ring	1.658870	0.75	2.903
The Matrix	1.599723	0.750001	2.799
WALL·E	1.079083	0.75	1.888
Iron Man	1.078170	0.744256	1.880
Star Wars: Episode IV - A New Hope	1.014846	0.750104	1.776
Interstellar	1.027390	0.728099	1.775
Up	1.181593	0.5	1.772
Inglorious Basterds	0.839903	0.74735	1.467
The Lion King	0.924965	0.5	1.387
Raiders of the Lost Ark	0.922755	0.5	1.384
Batman Begins	0.791694	0.746023	1.382
Apollo 13	0.773215	0.745056	1.349
Toy Story	0.883630	0.5	1.325
Star Wars: Episode V - The Empire Strikes Back	0.853180	0.5	1.279
Deadpool	0.845530	0.504959	1.272
Avatar	0.788088	0.50104	1.182
Star Wars: Episode VI - Return of the Jedi	0.671724	0.75	1.175
Pirates of the Caribbean: The Curse of the Black Pearl	0.769509	0.5	1.154
The Incredibles	0.768320	0.5	1.152
Terminator 2: Judgment Day	0.646399	0.749296	1.130
Shrek	0.735299	0.5	1.102
Star Trek	0.724870	0.496217	1.084
Gladiator	0.722488	0.5	1.083
The Martian	0.707388	0.500004	1.061

value 1 + EXS IMPORTANCE, and value 1 was determined expertly based on several experimental verifications.

The final list of 25 recommended movies is depicted in Table 17.

As shown in Table 17, the recommended system increased the evaluation of certain movies and their importance (position) in the final list of recommended movies for the user.

Our recommender system was fully implemented as a web system. Fig. 8 shows the main page of the system Predictory.

4. Results

This section describes the validation of the proposed system. To assess the quality of performed recommendations in the system, standard metrics were used: precision, recall and F1-measure.

A general definition of precision and recall metrics is

- Precision is the ratio of R_L to N.
- Recall is the ratio of R_L to R.

where N denotes the size of the recommendation list L, R_L denotes the number of relevant items that are included in L, and R denotes the total number of relevant items (Aggarwal, 2016; Falk, 2019; Symeonidis et al., 2009).

Precision defines the ability of the system to propose content that is relevant for a given user. It concerns a ratio of relevant recommendations with respect to all recommendations for the user. Precision can be calculated using the following:

$$\text{Precision} = \text{Correctly recommended content}/\text{Total recommended content}$$

where correctly recommended content is the number of relevant recommendations marked by the user as “correctly recommended”. Total recommended content is the number of all recommendations provided to the user.

Recall defines the ability of the system to provide the user with relevant content. It concerns the number of correct recommendations in a set of relevant recommendations, i.e., top recommendations of the system. Recall can be calculated using the following:

$$Recall = Correctly\ recommended\ content/Relevant\ content$$

where correctly recommended content is the number of recommendations marked as “correctly recommended”. Relevant content is a set of top recommendations based on user recommendations.

The F1-measure is then the harmonic mean between precision and recall:

$$F1 - measure = 2 * precision * recall/precision + recall$$

As the identification of correct (relevant) and incorrect (irrelevant) items in the list of recommended items is highly subjective, it is impossible to automate this process. If we wanted to perform the experimental verification on selected users included in the MovieLens dataset, we would be able to detect favorite and unpopular genres for each user and suggest a final list of 25 recommended movies. However, it would not be possible to obtain feedback for the recommended movies in real time (marked relevant and irrelevant to the user). Therefore, we selected a group of 17 users who tested our approach and performed the marking as relevant and irrelevant. The group consists of users aged 18 – 35 with various preferences of movies and genres. These users are of a similar type to the MovieLens dataset users – there are users

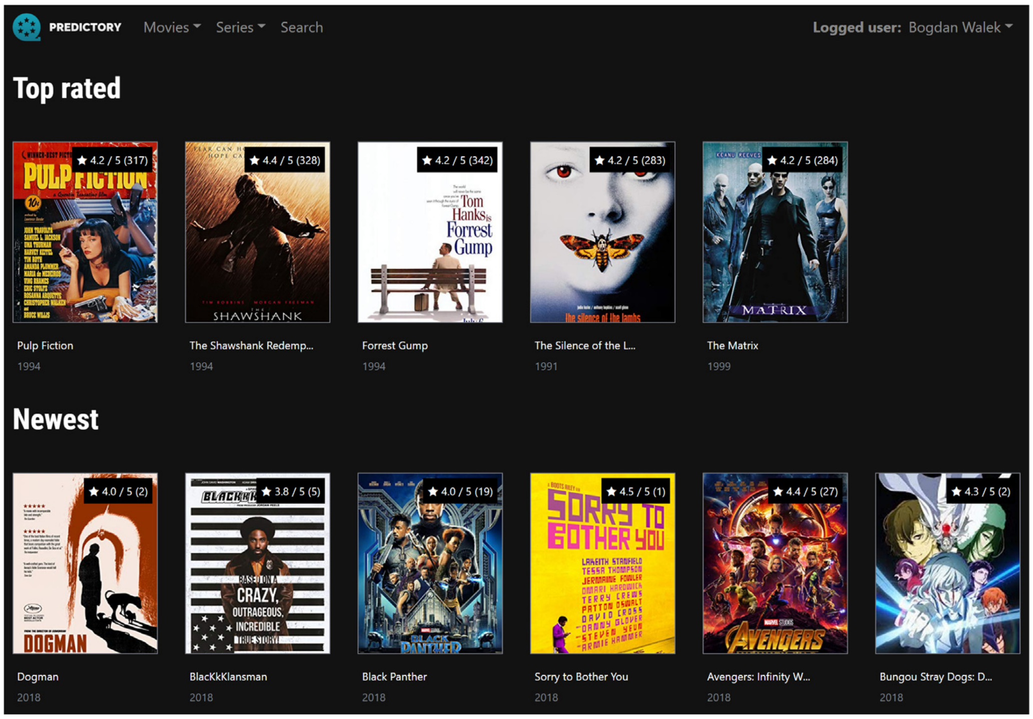

Fig. 8. Main page of the system Predictory.

who rated a large number of movies as well those who rated only a few – app. 20 movies. In total, it consists of 14% (86 out of 810) of the users of the MovieLens dataset. Testing on the group of real users is one of the possibilities that has also been used in other research projects (Barragáns-Martínez et al., 2015; Carrer-Neto et al., 2012; Colombo-Mendoza et al., 2015; Ho et al., 2006; Kumar et al., 2015; Li and Yamada, 2004). The users who tested our approach set their user preferences by inputting the following information:

- 20 interesting movies
- Rating of all 20 interesting movies using 0.5 star – 5 stars
- Identifying 3 favorite movie genres
- Identifying 3 unpopular movie genres

Then, a calculation of recommended movies for every user was performed on the basis of our proposed hybrid recommender system using an expert system. The system proposed 25 movies to every user, and the users marked either "correct" or "incorrect" movie recommendation – i.e., movie recommendation, if it concerns a relevant movie based on user preference or not.

The results are provided in Table 18. Precision signifies the ratio of movies marked as relevant to all recommended movies. Recall signifies the ratio of movies marked as relevant to the list of the top 15 recommended movies.

The results are also depicted in Fig. 9.

The results are promising. The system recommended a low number of relevant movies in five cases; in the remaining twelve, the number was high (20 - 25). Therefore, precision achieved a lower value in five cases (70%); in the other cases, the values were higher (80% - 100%). The recall metric denotes the movie ratio marked as relevant in the list of the top 15 recommended movies. In five cases, recall was lower (under 70%), and in the other cases, recall was higher (70% - 100%).

The diagram also shows that the precision average value in the test achieved 81%, recall 83%, and F1-measure 82%, which are promising values.

Table 18
Precision and recall metrics assessment.

User	Relevant	Irrelevant	Precision	Recall	F1-measure
1	16	9	64%	73%	68%
2	20	5	80%	67%	73%
3	12	13	48%	47%	47%
4	21	4	84%	93%	88%
5	25	0	100%	100%	100%
6	24	1	96%	100%	98%
7	24	1	96%	93%	95%
8	22	3	88%	100%	94%
9	22	3	88%	80%	84%
10	17	8	68%	60%	64%
11	25	0	100%	100%	100%
12	22	3	88%	93%	91%
13	22	3	88%	87%	87%
14	22	3	88%	100%	94%
15	21	4	84%	87%	85%
16	17	8	68%	67%	67%
17	13	12	52%	67%	59%
Total	**20**	**5**	**81%**	**83%**	**82%**

4.1. Comparison of our proposed approach with traditional approaches in the area of recommender systems

In this section, we compare our proposed approach with other traditional approaches to recommender systems to determine what results are achieved by our proposed system compared with other systems with respect to the ratio of relevant movies marked by users. The comparison was performed using the following:

- Collaborative filtering system – system using a pure SVD algorithm. The system was selected for its vast number of users and movie ratings, which are part of the MovieLens dataset.
- Content-based filtering system – a system using the TF-IDF algorithm. The system was selected for its large number of movies in the MovieLens dataset, which are potentially similar to the best-rated selected movies of the current user.

Table 19
Comparison of our proposed approach with other traditional approaches.

User	Content-based filtering	Collaborative filtering	Hybrid	Our system Predictory
1	4%	76%	68%	64%
2	0%	68%	60%	80%
3	24%	48%	44%	48%
4	40%	76%	88%	84%
5	72%	96%	96%	100%
6	60%	88%	76%	96%
7	56%	92%	88%	96%
8	4%	88%	72%	88%
9	64%	76%	76%	88%
10	68%	76%	80%	68%
11	72%	100%	88%	100%
12	24%	88%	84%	88%
13	60%	80%	84%	88%
14	84%	72%	76%	88%
15	48%	68%	68%	84%
16	20%	64%	48%	68%
17	24%	52%	48%	52%
	43%	**77%**	**73%**	**81%**

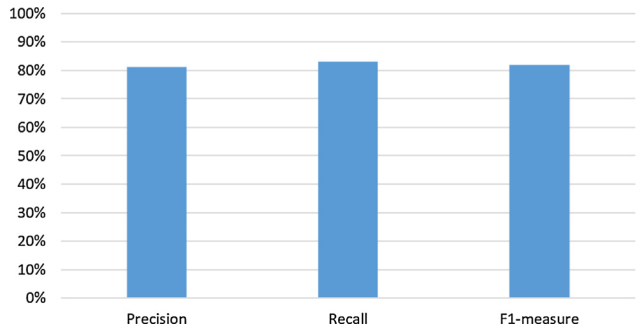

Fig. 9. Precision, recall, F1-measure metrics assessment.

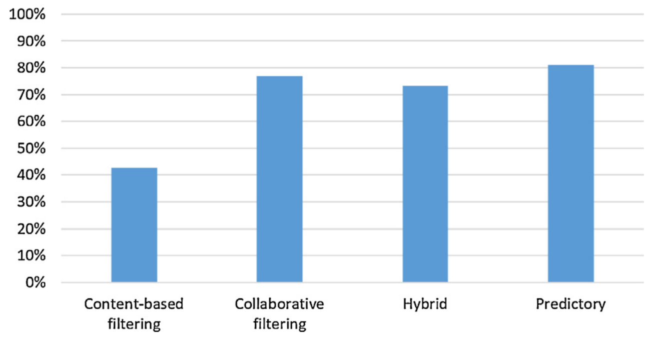

Fig. 10. Comparison of our proposed approach with other traditional approaches.

- Hybrid system – the system consists of a weigh-type hybrid system combining the results of the two previous systems. If the user did not add any rating, the user received the result of the content-based system; if fewer than 20 ratings, the result was composed of 20% of the collaborative filtering and 80% of the content based; if the user rated more than 20 items, the result comprised 80% of the collaborative filtering results and 20% of the content-based results.
- Predictory – a hybrid system with an expert system – our proposed recommender system.

The four systems were tested on the same group of users. Each system proposed the users 25 recommended movies. The main role of the users was to mark relevant and irrelevant movies in each system.

The ratio of marked relevant movies in all systems is depicted in Table 19.

The overall results are depicted in Fig. 10.

The results of the comparison are also promising. The content-based filtering system, which proposed movies based on similarity, achieved the worst score. Only 43% of the movies recommended by this system were relevant. The collaborative filtering system obtained much better results; 77% of movies were relevant. The hybrid system recommended 73% relevant movies. The best results were achieved by our proposed system; 81% of the recommended movies were marked by the users as relevant.

4.2. Comparison of our proposed system with other recommender systems

We also performed a comparison of our proposed approach with other open-source recommender systems. Open-source systems were selected to integrate the same MovieLens dataset used by our system. Thus, it is possible to compare the systems with respect to the relevance of the recommended movies (items). Integrating the same MovieLens dataset cannot be performed with the traditional commercial systems of YouTube or Netflix type; thus, we selected available open-source systems. The comparison included two suitable recommender systems: MovieGeek and Elastic Graph Recommender.

4.2.1. MovieGeek

MovieGeek is a recommender system incorporating a number of advanced algorithms used as a reference system (MovieGeek, 2019).

The system is written in Python (same as the analytical API of the proposed system) using the Django framework. The database system used in this system is PostgresSQL. MovieGeek includes both algorithms of collaborative filtering systems and content-based filtering systems. To recommend based on similarity to other users, it uses an approach focused on items. For a recommendation based on the similarity between individual items, it uses the LDA algorithm.

4.2.2. Elastic graph recommender

The Elastic Graph Recommender is a recommender system based on Python and then ElasticSearch and its module Elastic

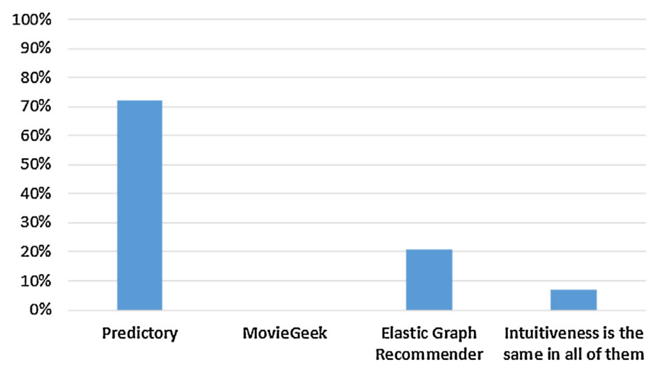

Fig. 11. Results of the comparison of the interface intuitiveness between the tested systems – Question 1.

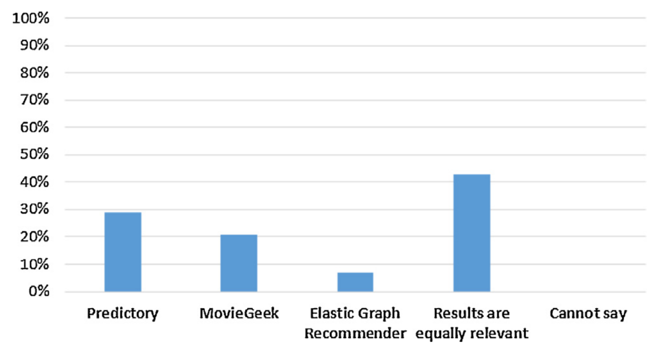

Fig. 12. Results of the comparison of the recommended movies relevance between the tested system – Question 2.

Graph, which enables the analysis and processing of dependencies of data, in this case, dependencies between users (Elastic Graph Recommender, 2019). This system implements only a recommendation based on the similarity between users. However, it also offers the possibility to refine the result based on the similarity of date of release, genre, or description.

The next step consisted of user testing of all the systems using respondents. They were expected to compare the systems while having the same input data and user preferences. The comparison concerned the intuitiveness of control, primarily based on the recommended movie relevancy. There were 14 respondents in the test. The respondents were independent users aged 18 – 35, and they were not part of the research team. The users went through individual movies in the systems, and in the case of the recommended movies relevance, they evaluated whether the system recommended suitable movies based on user preferences in the Recommended for you and You may also like sections.

Compared systems:

- Predictory (our proposed system)
- MovieGeek
- Elastic Graph Recommender

Respondents answered two questions:

- Question 1: Which of the offered systems offers the most intuitive interface?
- Question 2: Which of the offered system provides a recommended item that best matches your preferences?

The results of the test are depicted in Fig. 11 and Fig. 12.

The results of the comparison with other open-source systems are very promising. More than 71% of the respondents stated that our proposed system Predictory has the most intuitive interface, and 1 respondent stated that intuitiveness is equal for all. Only 21% of the respondents stated that intuitiveness is best for Elastic Graph Recommender. No respondent voted for the best interface in MovieGeek.

The second question was even more important as it concerned the ability to recommend relevant movies based on user preferences. Over 42% of the respondents (6 out of 14) stated that relevance is the same in all the systems. Over 28% of the respondents (4 out of 14) stated that our system gave most relevant results, 21% of the respondents (3 out of 14) stated that MovieGeek gave most relevant results, and 7% of the respondents (1 out of 14) stated that the Elastic Graph Recommender gave the most relevant results. The results show that most users think that recommendation relevance is the same in all systems.

However, it is necessary to emphasize that respondents' answers to both questions are subjective, so evaluation on a smaller set of test users is only indicative. Concerning the first question about the user interface, the level of subjectivity can play an important role. In addition, the intuitiveness of the interface does not belong to the main functionalities of the system. Concerning the second question, most respondents (42%) checked the answer "results are equally relevant", i.e., they could not decide which system offered the most relevant recommendation.

5. Conclusion

A comparison of individual recommender systems is quite a complex task, as they use various databases or databases with different amounts of data for validation (Rombouts and Verhoef, 2002). Nevertheless, the area of hybrid movie recommender systems has witnessed many systems described in the literature review (Véras et al., 2015). The advantage of existing hybrid systems is their effective ability to combine the content-based filtering approach and collaborative filtering approach, which enables movie prediction to be improved and reduces the cold-start and sparsity problems (Kumar et al., 2015; Lekakos and Caravelas, 2008; Rombouts and Verhoef, 2002; Salter and Antonopoulos, 2006; Symeonidis et al., 2009). The E-MRS system is interesting in its ability to propose suitable movies based on emotions. However, it was verified on a small number of users (Ho et al., 2006). Another perspective is recommender systems with explanations of why a given movie has been recommended; an example is MoviExplain (Symeonidis et al., 2009). This explanatory approach increases the credibility of the system and user loyalty (Aggarwal, 2016). However, a disadvantage of some hybrid systems is their complexity and robustness, which hampers their ability to recommend movies in real time due to the time complexity of computations (Aggarwal, 2016; Falk, 2019). The combination of collaborative filtering, content based, and knowledge-based systems is also limited. The authors in (Carrer-Neto et al., 2012) proposed a hybrid social knowledge-based system combining the hybrid approach with social networks, providing promising results based on experimental verification (high values of metrics precision, recall, F1-measure). A combination of a hybrid approach using social networks or explanations seems to be a more prospective direction in the future of hybrid system development (Aggarwal, 2016; Falk, 2019).

In this article, a monolithic hybrid system, Predictory, was proposed. The system combines a recommender module composed of a collaborative filtering system (using the SVD algorithm), a content-based system, and an expert system. An expert system was used to calculate the importance of individual items based on various parameters for the final evaluation of items in the list of recommended movies for a given user.

The construction of the recommender module takes advantage of a combination of collaborative filtering and content-based filtering systems. When predicting the evaluation within the collaborative filtering system, we worked with the favorite genres of the user. Movies having such a property have the highest predicted evaluation, which ranks them to higher positions in the list of recommended movies.

The resulting recommender system was validated and verified on a group of users using the MovieLens dataset with promising results. The proposed approach was also compared with other typical approaches. The results presented in this article have several practical implications:

- The system works with various user preferences (apart from favorite movies, with favorite and unpopular genres). Movies containing user favorite genres have a higher predicted evaluation. In contrast, movies with unpopular user genres have a lower predicted evaluation.
- The system uses a fuzzy expert system, which evaluates the level of importance of movies for the final list of recommended movies based on various parameters (average movie rating, number of movie ratings, level of similarity with already rated movies). IF-THEN rules of the expert system can be easily modified according to the need, and the whole expert system can be extended with other parameters in the future.
- It provides a functional combination of the collaborative filtering approach, content-based approach and a fuzzy expert system for the purposes of calculating the final list of recommended movies.
- Based on the results from the experimental verification, standard metrics achieved promising values: precision - 81%, recall - 83%, and F1-measure - 82%. In addition, when compared with other standard approaches (pure content-based system, pure collaborative filtering system, weighted hybrid system), our system achieved the highest ratio of relevant movies marked by the users during testing.
- Our proposed hybrid system using an expert system was fully implemented in a web application, Predictory. The system is completely available at https://app.predictory.dev. The system can thus be tested and verified for its functionality.

5.1. Future work

In our future work, we would like to focus on several areas. The first area is to work with favorite and unpopular movie genres. If a user sets “comedy” as an unpopular genre, the system will recommend other genres. However, user preferences can change over time. Thus, after some time, if the user begins giving good ratings in the “comedy” genre, the system could change the user’s preference for “comedy” from unpopular to favorite. It would allow the system to dynamically evaluate user preferences and automatically use them with genres set by the user together with genre preferences evaluated in real time based on user work in the system.

Another area is the integration of more rating and review platforms for the final movie evaluation. Our work with average movie ratings now uses movie ratings within the MovieLens dataset. The advantage is that, apart from an average movie rating, we also have all movie ratings related to real users. It enables us to easily create a user-item matrix to be used in the collaborative filtering system. However, current platforms (IMDb, Rotten Tomatoes, Metacritic, etc.) mostly work with aggregated average movie ratings, and it is impossible to obtain ratings of individual users within free accessible data of these services. Therefore, we want to modify the current fuzzy expert system in such a way that it would be able to work with average movie ratings from various platforms and thus evaluate movie importance for the final ranking and recommendation to a given user. Individual platforms would also be assigned different weights based on the number of users who rated the movie or based on other parameters.

We also want to determine other users’ preferences, such as favorite/unpopular actors and favorite/unpopular directors. Based on these preferences, it will be possible to modify the recommended system to consider these preferences. One method would be to weigh more movie genres with favorite actors directed by a favorite director. However, this might result in a conflict when there is one favorite and one unpopular actor in a given movie or more unpopular actors versus one favorite actor. System modification would thus be made based on experimental verification and other approaches to this area, e.g., the system (Carrer-Neto et al., 2012).

In addition, we would like to verify the proposed system in another problem domain, e.g., in the area of hotels and restaurants, for recommendations of suitable restaurants and hotel services based on various guest preferences.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Bogdan Walek: Conceptualization, Formal analysis, Methodology, Project administration, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. **Vladimir Fojtik:** Methodology, Software, Validation, Writing - review & editing.

Acknowledgement

The research was supported by a Student Grant SGS01/UVAFM/2019 with student participation, supported by the Czech Ministry of Education, Youth and Sports.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2020.113452.

References

Adomavicius, G., & Tuzhilin, A. (2005). Towards 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.

Aggarwal, C. C. (2016). *Recommender systems*: 1. Cham: Springer International Publishing.

Barragáns-Martínez, B., Costa-Montenegro, E., & Juncal-Martínez, J. (2015). Developing a recommender system in a consumer electronic device. *Expert Systems with Applications, 42*(9), 4216–4228.

Burke, R. (2007). *Hybrid web recommender systems. In the adaptive web (pp. 377-408)*. Berlin, Heidelberg: Springer.

Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R., & García-Sánchez, F. (2012). Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications, 39*(12), 10990–11000.

Cheng, Z., & Shen, J. (2016). On effective location-aware music recommendation. *ACM Transactions on Information Systems (TOIS), 34*(2), 13.

Cheung, K. W., Kwok, J. T., Law, M. H., & Tsui, K. C. (2003). Mining customer product ratings for personalized marketing. *Decision Support Systems, 35*(2), 231–243.

Colombo-Mendoza, L. O., Valencia-García, R., Rodríguez-González, A., Alor-Hernández, G., & Samper-Zapater, J. J. (2015). RecomMetz: a context-aware knowledge-based mobile recommender system for movie showtimes. *Expert Systems with Applications, 42*(3), 1202–1222.

Elastic Graph Recommender, (2019)., https://github.com/o19s/elastic-graph-recommender

Elmisery, A. M., & Botvich, D. (2011). Agent based middleware for private data mashup in IPTV recommender services. In *In 2011 IEEE 16th international workshop on computer aided modeling and design of communication links and networks (CAMAD)* (pp. 107–111). IEEE.

Falk, K. (2019). *Practical recommender systems* (1st ed.). Shelter Island: Manning ISBN 1-61729-270-2.

Fortune, Amazon’s recommendation secret, (2012)., https://fortune.com/2012/07/30/amazons-recommendation-secret/

Ghani, R., & Fano, A. (2002). Building recommender systems using a knowledge base of product semantics. In *In Proceedings of the workshop on recommendation and personalization in ECommerce at the 2nd international conference on adaptive hypermedia and adaptive web based systems*. Citeseer.

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.

Habiballa, H., Novák, V., Dvořák, A., & Pavliska, V. (2003). Using software package LFLC 2000. In *2nd International Conference Aplimat 2003* (pp. 355–358).

Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: history and context. *Acm transactions on interactive intelligent systems (tiis), 5*(4), 19.

Herrera-Viedma, E., Porcel, C., Lopez-Herrera, A. G., & Alonso, S. (2008). A fuzzy linguistic recommender system to advice research resources in university digital libraries. In H. Bustince, F. Herrera, & J. Montero (Eds.). In *Fuzzy sets and their extensions: representation, aggregation and models: 220* (pp. 567–585). Heidelberg: Springer.

Ho, A. T., Menezes, I. L., & Tagmouti, Y. (2006). E-mrs: emotion-based movie recommender system. In *Proceedings of IADIS e-commerce conference* (pp. 1–8). University of Washington Both-ell.

Katarya, R., & Verma, O. P. (2017). An effective collaborative movie recommender system with cuckoo search. *Egyptian Informatics Journal, 18*(2), 105–112.

Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender systems: issues, challenges, and research opportunities. In *In Information Science and Applications (ICISA) 2016* (pp. 1179–1189). Singapore: Springer.

Kumar, M., Yadav, D. K., Singh, A., & Gupta, V. K. (2015). A movie recommender system: Movrec. *International Journal of Computer Applications, 124*(3).

Lekakos, G., & Caravelas, P. (2008). A hybrid approach for movie recommendation. *Multimedia Tools and Applications, 36*(1–2), 55–70.

Lester, P. M. (2013). *Digital innovations for mass communications: Engaging the user*. Routledge.

Li, P., & Yamada, S. (2004). A movie recommender system based on inductive learning. In *In IEEE conference on cybernetics and intelligent systems, 2004: 1* (pp. 318–323). IEEE.

Logesh, R., Subramaniyaswamy, V., & Vijayakumar, V. (2018). A personalised travel recommender system utilising social network profile and accurate GPS data. *Electronic Government, an International Journal, 14*(1), 90–113.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support System, 74*, 12–32.

Martinez, L., Barranco, M. J., Perez, L. G., & Espinilla, M. (2008). A knowledge based recommender system with multi granular linguistic information. *International Journal of Computer Intelligent System, 1*(3), 225–236.

MovieGeek, (2019)., http://www.moviegeek.eu/

MovieLens, (2019)., https://movielens.org/

Ojokoh, B., Omisore, M., Samuel, O., & Ogunniyi, T. (2012). A fuzzy logic based personalized recommender system. *International Journal of Computer Sccence of Information Technology Security, 2*(5), 1008–1015.

OMDb API – The Open Movie Database, (2019)., http://www.omdbapi.com/

Oramas, S., Ostuni, V. C., Noia, T. D., Serra, X., & Sciascio, E. D. (2017). Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST), 8*(2), 21.

Parra, D., & Amatriain, X. (2011). Walk the talk. In Joseph A. Konstan, R. Conejo, José L. Marzo, & N. Oliver (Eds.). In *UMAP 2011. LNCS: 6787* (pp. 255–268). Heidelberg: Springer. https://doi.org/10.1007/978-3-642-22362-4_22.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *In The adaptive web* (pp. 325–341). Berlin, Heidelberg: Springer.

Perny, P., & Zucker, J. D. (2001). Preference-based search and machine learning for collaborative filtering: the "film-conseil" movie recommender system. *Information, Interaction, Intelligence, 1*(1), 9–48.

Ponnam, L. T., Punyasamudram, S. D., Nallagulla, S. N., & Yellamati, S. (2016). Movie recommender system using item based collaborative filtering technique. In *In 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)* (pp. 1–5). IEEE.

Ravi, L., & Vairavasundaram, S. (2016). A collaborative location based travel recommendation system through enhanced rating prediction for the group of users. *Computational Intelligence and Neuroscience, 2016*, 7.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook*. Heidelberg: Springer.

Rodríguez-García, M. Á., Colombo-Mendoza, L. O., Valencia-García, R., Lopez-Lorca, A. A., & Beydoun, G. (2015). Ontology-based music recommender system. In *Distributed computing and artificial intelligence, 12th international conference Springer, Cham* (pp. 39–46).

Rombouts, J., & Verhoef, T. (2002). A simple hybrid movie recommender system.

Salter, J., & Antonopoulos, N. (2006). CinemaScreen recommender agent: combining collaborative and content-based filtering. *IEEE Intelligent Systems, 21*(1), 35–41.

Symeonidis, P., Nanopoulos, A., & Manolopoulos, Y. (2009). MoviExplain: a recommender system with explanations. In *In Proceedings of the third ACM conference on Recommender systems* (pp. 317–320).

Stanley, L., Lorenzi, F., Saldaña, R., & Licthnow, D. (2003). A tourism recommender system based on collaboration and text analysis. *Information Technology and Tourism, 6*(3), 157–165.

Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A., & Vijayakumar, V. (2017). A personalised movie recommendation system based on collaborative filtering. *International Journal of High Performance Computing and Networking, 10*(1–2), 54–63.

TAMMA Capital, LLC., Amazon's recommendation secret, (2019)., https://tammacapital.com/amazons-recommendation-secret/

Véras, D., Prota, T., Bispo, A., Prudêncio, R., & Ferraz, C. (2015). A literature review of recommender systems in the television domain. *Expert Systems with Applications, 42*(22), 9046–9076.

Wu, D., Zhang, G., & Lu, J. (2015). A fuzzy preference tree-based recommender system for personalized business-to-business e-services. *IEEE Transformation of Fuzzy System, 23*(1), 29–43.

Xiao, B., & Benbasat, I. (2014). Research on the use, characteristics, and impact of e-commerce product recommendation agents: a review and update for 2007–2012. In *Handbook of Strategic e-Business Management* (pp. 403–431). Springer.

Yager, R. R. (2003). Fuzzy logic methods in recommender systems. *Fuzzy Sets System, 136*(2), 133–149.

Zenebe, A., & Norcio, A. F. (2009). Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems. *Fuzzy Sets System, 160*(1), 76–94.

Zhang, Z., Lin, H., Liu, K., Wu, D., Zhang, G., & Lu, J. (2013). A hybrid fuzzy-based personalized recommender system for telecom products/services. *Information Science, 235*, 117–129.