An Intelligent System of the Content Relevance at the Example of Films According to User Needs

Vasyl Lytvyn[0000-0002-9676-0180]1, Aleksandr Gozhyj[0000-0002-3517-580X]2, Irina Kalinina[0000-0001-8359-2045]3, Victoria Vysotska[0000-0001-6417-3689]4, Viktor Shatskykh[0000-0002-9497-0256]5, Lyubomyr Chyrun[0000-0002-9448-1751]6 and Yuriy Borzov[0000-0002-0604-0498]7

[1,4-5]Lviv Polytechnic National University, Lviv, Ukraine
[2-3]Petro Mohila Black Sea National University, Nikolaev, Ukraine
[6]Ivan Franko National University of Lviv, Lviv, Ukraine
[7]Lviv State University of Life Safety, Lviv, Ukraine
vasyl.v.lytvyn@lpnu.ua, alex.gozhyj@gmail.com, irina.kalinina1612@gmail.com, Victoria.A.Vysotska@lpnu.ua

Abstract. In the work the methods of forming recommendations to user are analyzed, an overview of existing recommender systems is implemented. The object of research is the process of providing relevant recommendations for the selection of films to the user. The subject of research is the methods and means of providing relevant film recommendations according to needs of user. An algorithm for forming films recommendations based on collaborative filtration has been improved. Practical result of work is developed recommender system for selecting film based on the proposed algorithm as web-application form. It is recommended that the system be used for data collection about preferences of different people in the selection of films and provide appropriate relevant recommendations. In further research in this area, it is advisable to improve the way to store movie data and replace the ranking algorithm of movies based on reviews, grounded on such algorithm that takes into attention the qualitative and quantitative characteristics of reviews.

Keywords: Commercial Content, Personalization, Machine Learning, SEO Technology, Search Metrics, E-Commerce, NLP, Recommender System, Recommendations, Collaborative Filtering, Content Filtering, Web Application.

1 Introduction

Amount of information worldwide is growing at a rapid rate over a considerable period. People daily acquire and filter the incoming stream of information coming from different sources: work, social networks, e-mail, online cinemas, popular sources of information, etc. After invention of the Internet, amount of such information has grown rapidly, a large number of services have appeared to provide users with everything they need for a comfortable life. In recent years, Internet services that offer products of all possible types (online stores), information for everything (Internet magazines, films, news, forums, books, serials, articles) and so on have become very popular. Its became extremely difficult for a user to navigate in movie catalogs and serials, even with built-in search and filtering, since it's very difficult to make a choice

with such a large amount of information. The World Wide Web allows you to collect conveniently information about the preferences of users of online sources. The World Wide Web has a large number of search engines, directories, ratings and file sharing systems, so the amount of content is constantly increasing, as the each site or service provides information that, in the process of its operating, generates even more volumes of information. The main, but not the only task of recommender services is to help the user of content to navigate in the search results. Generally, recommendations for films from various online services of the existing recommender systems are not always relevant to users, as often based on unproductive and archaic algorithms. There are many problems with creating a prediction system for movie recommendations. Therefore, it is very interesting to see, how the different approaches to creating recommender systems solve these problems and improve the quality of the recommendations. The topic of scientific work is relevant in Ukraine, since there are no proper services on the Internet to provide relevant film recommendations in Ukrainian. The purpose of current scientific work is:

- An analysis of the existing film recommender systems,
- Research of recommendation algorithms and identification of their advantages and disadvantages,
- Development of algorithms for the formation of recommendations on the basis of collaborative filtration,
- Description of the operating principles of the recommender system for the selection of films,
- Practical implementation of the information recommender system for selection relevant films according to needs of user based on the developed algorithm.

2 Analysis of the Methods of Forming Recommendations in Recommender Systems

Nowadays, with growing of a large number of diverse goods and services, it is increasingly difficult to make a choice. There are new labor products that must find their buyer. Recommender Systems (RS) are software tools that try to predict which objects (movies, music, books, etc.) will be interesting to user if there is some information about his preferences [1-9]. Recommender system is a system that recommends goods to users among a huge stream of information, depending on their needs. Goods are elements of a specific service to which the user has an interest: films, restaurants, books, articles, etc. User interests can be represented in several ways: using ratings that users provide to products or with the keywords of each product [10-17]. Recommender systems change the way websites, software applications interact with their users. Instead of providing static information, when users search and possibly buy products, recommender systems increase the level of interactivity, expanding the capabilities presented to user. Recommender systems form recommendations for each client based on the history of his purchases or searches, and on the behavior of other users. Most f large-scale commercial and social websites offer their users a variety of suggestions, such as products for future study or people to whom it is appropriate to

refer. Recommender mechanisms sort large volumes of data to identify potential users' preferences [19-28]. To maintain user preferences for products, RS use user profiles. In most RS, the user profile contains sets of ratings and / or keywords (tags). Rates provided by users to goods may belong to different ranges (0-1, 1-5, 1-10): the higher the rating, the more specific product liked the user. After each evaluation, all user ratings are aggregating through a number of calculations, measuring the similarity of users, and then predict recommendations for this user. Keywords are automatically loading from texts or goods that users viewed or evaluated in the past. They can also be important depending on how the user has evaluated a particular word. More words that are meaningful will be more important than less significant (TF-IDF algorithm). After that, the texts (goods) are mapping to the user profile and the most appropriate ones are recommended [29-36]. Ratings can be explicit and implicit. An explicit rating is an estimate by which the user showed interest in that product within its own rating system. Implicit ratings are calculating from the purchase history or user behavior. Their advantages include the reduction of the load on the user's valuation of goods. The source of implicit ratings may be the time spent reading an article, the reference to the product in other sources (for example, the algorithm for ranking the pages of the Google search engine). Other indicators of the review behavior, such as cursor movement, keyboard input and page scrolling speed, were explored also as implicit indicators of interest and demonstrated good results [37-45]. The general architecture of the recommender system (Fig. 1) can be represented as follows:

1. Background data is system information about the products of service, which is usually collects in the background [46-54].
2. Input data is the information the user enters into the system to get recommendations [55-63].
3. The recommender algorithm is an algorithm that combines system information and input data to obtain recommendations [64-73].

Such systems as reference data use the user profiles, and as inputs - user actions (product evaluation, time spent on a page, etc.) (Fig. 1) [2, 74-78].

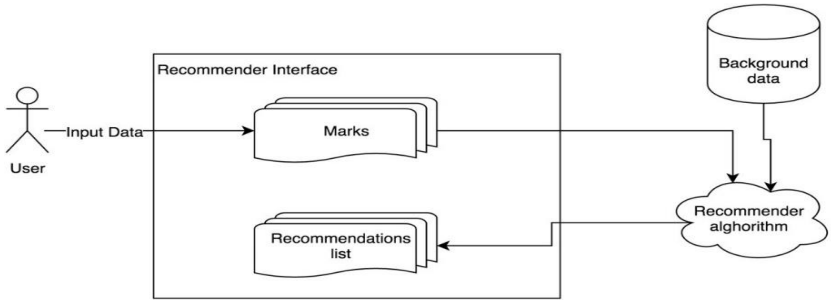

Fig. 1. Basic architecture of recommender system

There are four main types of recommender systems [3, 79-85]: content-based; collaborative; knowledge-based; hybrid. Most of recommender systems use one of two

basic approaches: collaborative filtering or content filtering. There is also a class of approaches based on the combination of these two basic - hybrid filtering.

Below are the general steps to formulate recommendations (Fig. 2) [3]:

1. Preprocessing, which includes finding data similarity, sampling, dimension of data;
2. Data analysis that includes classification and clustering of data;
3. Presentation of results.

The introduction of recommender systems expands the ability to interact with clients in various subject areas. The application of recommender systems leads to increasing sales of goods, sales expansion of more diverse objects, increase user loyalty; better understanding of user's needs and wishes. Consider the main types of recommender systems [86-89].

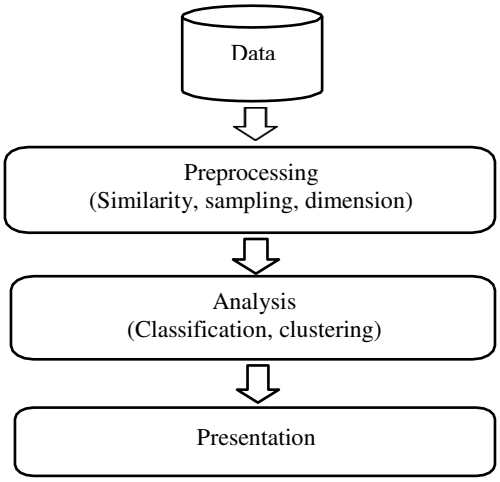

Fig. 2. General steps to formulate recommendations

2.1 Methods of content-based filtration

Recommender content filtering systems build internal links between objects that are liable to recommendation. Content-based filtration generates a recommendation based on objects rated by the user in the past. For example, if a user bought a book for guitar playing in a bookstore, than other popular books for self-study or the same author's book would automatically be offered to him [4].

The main steps in the recommended content filtering system: analyze the content of objects; form a set of its criteria's (genres, tags, words). Find out what criteria a user likes, compare these data, and get recommendations. Criteria's combine users and objects in one coordinate system - if the user's point and object are nearby, the user probably become interested in that object [4]. This approach can use retrospective information about view. For example, if any catering customer orders meals exclusively without meat content, the content filtering can use this retrospective information to identify similar dishes and recommend them to this customer.

The advantages of this approach are [4]:

- Independence from other users: the recommendations are formed on the basis of a particular user, and do not take into account the information about the preferences of other clients of the system;
- Transparence;
- Simplicity of adding new objects;
- Minimum information: only user and object data is required to get recommendations.

The disadvantages of the content-based filter approach are:

- Limited features of the content analyzer;
- Recommendations for new areas;
- Lack of data for new users.

2.2 Methods of collaborative filtering

Collaborative filtering systems formulate recommendations based on customer behavior information in the past, such as purchases or ratings. This approach can be built exceptionally on the basis of the behavior of this user or taking into attention the behavior of other users with similar characteristics that is more effective [4]. If, the collaborative filtering takes into account the behavior of other users, it uses group knowledge to formulate recommendations based on the similarity of users.

For example, during creation of a web blog that requires a recommender system based on information from many users that subscribe and read the blogs, these users can be grouped according to their interests. You can combine into one group the users who read several identical blogs and by this information identify the most popular blogs among them. The specific user from this group will be recommended the most popular blogs among those he have not been subscribed yet [5].

At first, an analysis of the information of a particular user and other clients of the system is conducted in order to identify the similarity of the characteristics. The result of the analyzer is "neighbors" - a group of users with similar preferences. Based on objects liked by "neighbors", recommendations for this client are formed.

There are a number of collaborative filtering algorithms (Fig. 3).

- Memory-based - methods that are based on an analysis of existing user ratings.
- Methods based on the analysis of the data model - Model-based methods.
- Methods based on the combination of previous algorithms - Hybrid methods.

Methods of closest neighbors are divided by the similarity analysis: User-based and Item based. The goal of both directions is to allocate similar objects to groups based on the matrix of evaluations. In the case of User-based, the similarity of users that previous ratings is similar to current user is determined, and estimates from other users are used to predict the preferences of this user. Another approach, Item based (based on the similarity of the elements), is used by Amazon.com. Instead of using the similarity between the behaviors of user estimates for the prediction, it uses the similarity between estimates of the model of elements. Usually, if two elements have the

same user ratings, they are similar, and users are probably having similar preferences for such items [6].
The following approaches can be used to determine the similarity between users or objects:

- Distance of Euclid, Hamming;
- Pearson correlation;
- Spearman rank correlation;
- Jacquard coefficient;
- Cosine similarity.

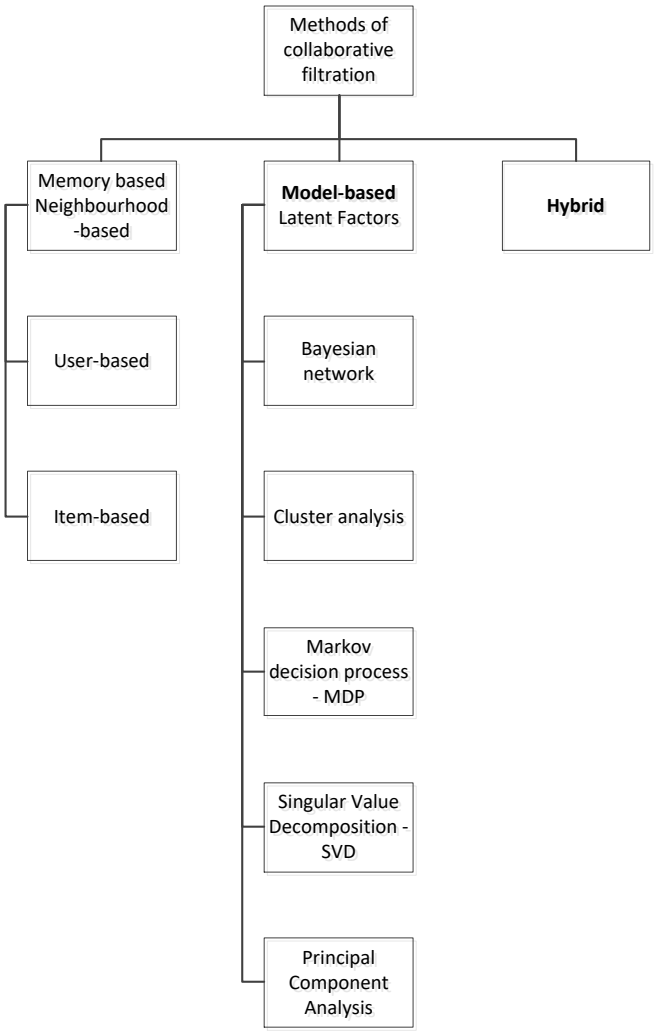

Fig. 3. Classification of collaborative filtration methods

Collaborative filtering based on the similarity of users has a high accuracy. However, the disadvantage of this approach is high resource intensities (memory requirements) and complexity (the number of computations required to obtain a recommendation). In addition, the calculation of similarity measure can be deducted only in real time, since the current transaction data is only available now of the calculating recommendation. Therefore, this approach can be applied to relatively small databases [7]. In the item-based similarity algorithm, the measure of intimacy of an element to all others can be calculated in a separate mode according to the schedule, as the rating vectors of all elements are available up to the moment of forming the recommendation. Thus, this algorithm is more effective in terms of time of forming recommendations, thanks to the possibility of a separate preprocessing of data.

The information area for collaborative filtering systems consists of users who liked different objects. Preference (estimate) is often gives in the form of a triplet (user, object, score). These estimates may take on different forms, which, on their part, depend on the subject matter of the recommender system applying. Collaborative filtering - the most popular type of algorithms today for creating recommender systems. The main advantages of this approach for forming of recommendations are [4]:

- Simplicity and speed of algorithms (K-based, etc.);
- Simplicity of explanation;
- Stability.

The main disadvantages of collaborative filtering are the unresolved “cold start” problems - the lack of data on recently introduced users, the "fraud", the scarcity of the matrix of estimates (sometimes impossible to predict) that complicates the forming of recommendations.

2.3 Hybrid methods of forming recommendations

Hybrid approaches combine different recommender mechanisms. This increases the accuracy of recommendations and increases the complexity of recommender systems. The most common combination is a joining the results of collaborative and content filtering. A hybrid approach may be useful if the application of collaborative filtration begins with a significant data thinner. The hybrid approach allows you first to weigh the results according to the content filtering, and then shift those weights toward collaborative filtering (depending on the degree of “ripeness” of the available dataset for a particular user) [8]. An example of a hybrid recommender system is Netflix, which combines 27 recommendation algorithms [9]. The application of hybrid methods to forming recommendations increases the complexity, and the cost of the system. The successful creation and implementation of the hybrid algorithm requires a team of experienced developers who have a number of professional skills.

2.4 An analysis of known problem solving tools

MovieLens is an American recommender web system that uses collaborative filtering and other algorithms for movie recommendations. The system was created at the University of Minnesota for research purposes. The service offers the user to choose the favorite genres and movies, and then tries to predict the ratings of popular films and

novelties for current user that must correct them on his side. After this, the service provides the user with appropriate recommendations.

MovieLens builds its recommendations on materials provided by site users, including movie ratings. Now the service has more than 20 million estimates [10]. The site uses various algorithms for recommendations, including collaborative filtering algorithms such as item-item, user-user, and SVD. In addition to solving the "cold start" problem for new users, MovieLens uses methods to collect information about the benefits of users. The system asks new users to rate how much they like to watch different movie groups (for example, movies with black humor or romantic comedies). The preferences recorded with this poll allow the system to make initial recommendations even before the user has evaluated a large number of movies on the site. For each user, MovieLens predicts how he will rate any movie on the site. Based on these predicted ratings, the system recommends movies that are likely to be highly rated by the user. However, the MovieLens rating approach is not always effective. Researchers have found that over 20% of the films listed in the system have so few estimates that recommendation algorithms cannot make accurate predictions about whether users will like these films or not [10]. In addition to the recommendations, MovieLens also provides information about individual movies, such as the list of actors and directors of each movie. The advantage of MovieLens is the fine-tune-up of finding movies that are similar to those you prefer. You can reduce or increase the proposed criteria, or set your own. The disadvantage of MovieLens is the lack of movie description in Ukrainian. Therefore, those who do not know the English, the details will need to be specified apart.

Megogo is a Ukrainian video service that allows legally to view licensed movies online at Full HD (1080p), changing the language of audio tracks and subtitles. There is also a possibility to watch popular television channels. Besides the general options for choosing a movie: year, genre and country, the service offers smart recommendations based on preferences (at least five films must be rated) and thematic collections. Lack of resource is not sufficiently relevant recommendations. Advantages are a simple and friendly interface in Ukrainian and availability ofspecial section with thousands of films for hearing impaired people [11].

3 Software to Solve the Problem

3.1 Choice and justification of problem solving software

After analyzing such systems, it was decided to implement the system as a web application. To implement the server part of the application, you could choose the following programming languages, such as Python or Ruby. However, it was decided to use the programming language R-language, which is a powerful data analysis environment, and it is most suitable for bulk statistical computing and data visualization, which is an important factor in the development of recommender systems, as the movie databases and their ratings are very large in scope and require to be analyzed.

3.2 Specifications of selected software development tools

Several graphical interfaces and development environments can be used to implement the project. There are following among them: RStudio, RKward, RapidMiner R, Java Gui for R (JGR), Deducer, Rattle GUI, R Commander, R Development Tools (Visual Studio), StatET (Eclipse), etc.

RStudio [13] is a free integrated development environment (IDE) for R-language. Thanks to its features, this software product is actively developing and provides Joseph Allayre, the creator of ColdFusion (programming language), launched convenient work with R. RStudio. Chief Scientist at RStudio is Hadley Wicham. RStudio is available in the following versions: RStudio Desktop and RStudio Server. In Desktop version, the program can be compared to ordinary applications. The Server also provides access to RStudio that runs on a remote Linux server using the web interface. Desktop version distributions are available for Windows, GNU Linux, and OS X. RStudio is available in two editions - free and commercial. It is capable of running on computers (Windows, Linux, and OS X) or in any browser that is connected to RStudio Server version or RStudio Server Pro version (Ubuntu, Debian, Red Hat Linux, CentOS, SLES and openSUSE).) [13]. RStudio was written using the C ++ language and the Qt framework for the UI (user interface). RStudio began developing in December 2010. For the first time, the public beta-version was announced at the official level in February 2011. Version 1.0 was released on November 1, 2016.

RStudio IDE was chosen to develop the whole system, as for today it is the most optimized for the development of R-applications. RStudio allows you to install quickly any packages from CRAN with one command in the console. An important argument in favor of RStudio is the ability of quick download of local version of the Shiny-app to the server using only one button in the interface that highly simplifies the upgrade to a newer version [13]. A free subscription to shinyapps.io has been selected to expand the web application, which includes up to five active applications at a time and 25 active hours per month for each application. This service has an informative dashboard with current information about using the web application.

4 Description of the created software

As a result of practical part of the work, the system was developed as a web-based application "VikToFilm" - a recommender system of films based on the User-based Collaborative Filtering algorithm. The entire “back end” is written in the programming R-language of version 3.4.0, and the client part, the “front end”, is created using the Bootstrap 3.3.6 and the Shiny 1.0.2 framework. As this system is a web application developed in the programming R-language using the Shiny framework, the software necessary for its operation is shin yapps web server [14] with the application expanded on it. The Shiny service works in a cloud on servers owned by RStudio Corporation. Each application is autonomous and works with any data downloaded with the application or data extracted from third-party data stores such as databases or web services. In order to expand the application on the Shiny Web servers, you need a locally installed language development environment R version 3.2 or higher, such as RStudio IDE. To install and use RStudio IDE version 1.0.136, any computer running

Linux/Unix, Windows, or Mac OS X with at least 1 gigabytes of RAM and a 2 GHz processor can be used. You also need to have CRAN installed - an archive of network packets for the programming R-language, an R-packet reconnect for connecting to Shiny servers and expanding the project to them. In addition, you must create a shinyapps.io user account to manage the server part of the web application. The created system solves the tasks related to the prediction of film recommendations, the choice of films, which will be relevant to each individual user based on his preferences. This system is recommended for collecting user preferences during movies selection and providing appropriate relevant recommendations. The algorithm of the program can also be adapted for the recommendations of series, music, books, restaurants, etc. The application is based on the collaborative filtration algorithm. The basic idea of CFR (Collaborative Filtering Recommender) is that, if two users had the same interests in the past, for example, they prefer the same movie, and then these users will have similar flavors in the future. If, for example, user A and user B have a similar story of movie views and user A recently reviewed and highly rated the movie that user B has not seen, then this movie will be offered to user B. The joint filtering approach takes into account only the user's preferences and does not take into attention the features of objects (for example, movies) to determine the recommendations. In the designed project, in order to recommend films, it was decided to use the user preferences during film selection. Data for processing was taken from MovieLens [15]. Here the films are rated by 5-point scale. In order not to overload the system with large calculations, the project used the smallest available version (ml-latest-small.zip), which at the time of loading contained **100 004** ratings and 1 296 keywords for **9 125** films. The 671 user between January 9, 1995 and October 16, 2016 created this data. This dataset was created on **October 17, 2016**. Users were chosen randomly. All selected users have already rated at least **20** films. The processing data is contained in four files: links.csv, movies.csv, ratings.csv and tags.csv. The created system uses only **movies.csv** and **ratings.csv** files to build its own system of recommendations. In CSV-files, the fields are separated by comma and new line character.

The process of data preparation consists of the following steps:

1. Select relevant data.
2. Normalize data.
3. Convert data into binary format.

In order to select the most relevant data, the minimum number of users who rated the movie as 50 and the minimum number of movie views as 50 (Fig. 4) was determined.

```
421 x 444 rating matrix of class 'realRatingMatrix' with 37915 ratings.
```

Fig. 4. Minimum number of relevant users and movies

Such sample of the most relevant data contains **421** user and **444** movies (compared to the previous 671 user and 9066 movies in the general data set). Using the same approach as before, the top 2 percent of users and movies in the new matrix of the most relevant data were visualized (Fig. 5-6).

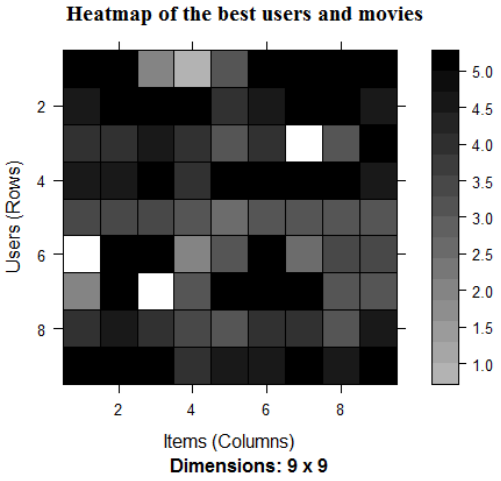

Fig. 5. Heat map of the best users and movies

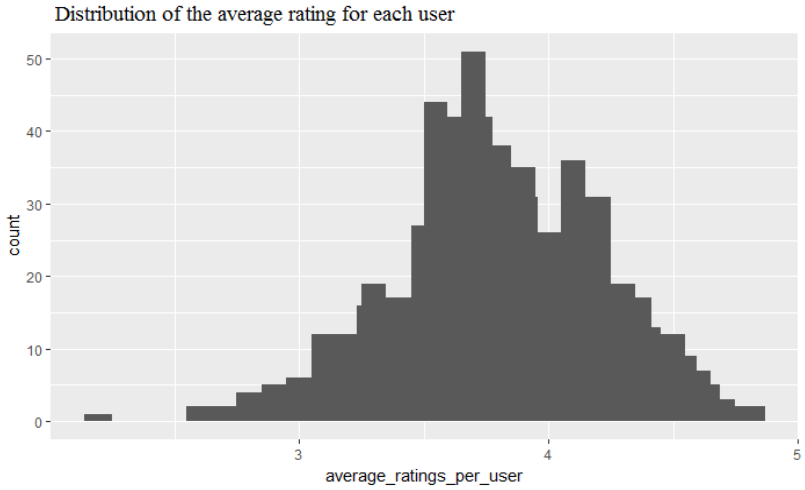

Fig. 6. Distribution of the average rating for each user

On the heat map some lines are darker than others. This may mean that individual users give higher ratings to all movies. The distribution of the average rating for each user in comparison with others is very variable, as shown in Fig. 6. Users who only give high (or only low) ratings to all viewed movies can spoil the results. To avoid this, the data was normalized so that the average score of each user was zero. For quick checking, an average rating of each user was calculated equal to zero (Fig.7).

```
> ratings_movies_norm <- normalize(ratings_movies)
> sum(rowMeans(ratings_movies_norm) > 0.00001)
[1] 0
```

Fig. 7. Data normalization

The next step is to visualize a normalized matrix for top films. As the data is continuous, the heatmap is in color tones (Fig. 8).

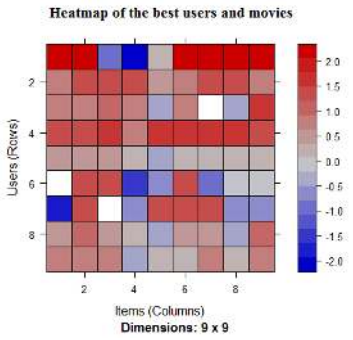

Fig. 8. Heatmap of normalized the best users and movies

In Fig. 8 depicts several lines that appear to be more bluish or redder. The reason is that only the best movies were visualized. The average rating is zero for each user.

Some recommendation models work on binary data, so it will be useful to convert the data to a binary format, that is, to define a table that contains only 0 and 1. Missed values or poor rates can be considered as zero. In this case, it is expedient:

- Define a matrix that has a value or one, if the user evaluated the movie, or 0, otherwise. In the last case, rating information is lost.
- Define a matrix with one, if the score is higher or equal to a certain threshold (for example, 3), and zero otherwise. In this case, giving bad rate to movie means not to appreciate it at all.

The expediency of choosing one or another method depends on the context. Using two different approaches, two matrices were determined, and a 5 percent part of each of the binary matrices was visualized. In the first version, it is necessary to determine a matrix, whose cells are equal to one, if the film was reviewed. In the second version, you need to determine matrix, whose cells are equal to one, if the cell has a rating above the threshold value.

The following user-based approach was used. According to this approach, the new user is looking for similar users. The movies that these users rated high are recommended to the new user [16]. For each new user, the following steps are performed:

1. Determination of the degree of similarity of each user-neighbor to the new one. As in the IBCF, the cosine factor or Pearson correlation coefficient is used.
2. Determination of the most similar users. Options:

- Using top k users (k nearest neighbors).
- Using users whose similarity is higher than a certain threshold.

3. Evaluation of movies according to the rating given by the most similar users. Final rating is an average rating among similar users. Use the following approaches:
- Average rating.
- Weighted average rating, using similarity as weights.
4. Getting the Most Relevant Movies.

At first, check the default settings for the UBCF model. Here *n* is the number of similar users. The similarity function by default uses cosine coefficient - **cosine**. Therefore, it is worth building a model using standard settings and a training set of data. As with IBCF, ten best recommendations have been identified for each new user. The matrix above (Fig. 9) contains the movieId of each recommended movie (rows) for the first four users (columns) of the entire dataset.

```
      [,1] [,2] [,3] [,4]
 [1,] 1089  608  858 1090
 [2,] 1206  318  260  595
 [3,]  293  527  318 1080
 [4,]  858   47  899 2804
 [5,]   47  593   50 1278
 [6,] 1732 2959  246 1674
 [7,] 1213   50  111 2529
 [8,] 6016 2571  750  596
 [9,]  924  223 1198  661
[10,]  111  923  923 1285
```

Fig. 9. Matrix with recommendations for the first 4 users (UBCF)

It was also calculated, how many times each film is recommended and the corresponding frequency histogram is constructed (Fig. 10):

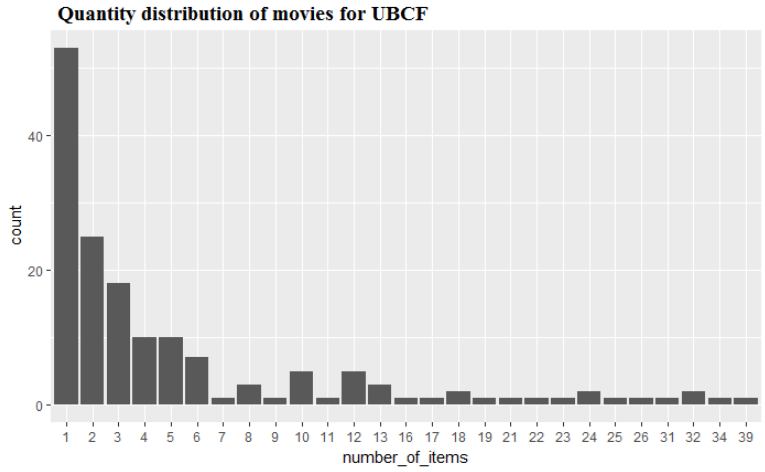

Fig. 10. Quantity distribution of movies for UBCF

Compared to IBCF, the distribution has a longer tail. This means that some movies are recommended more often than others. The maximum is more than 30, compared to 10 for the IBCF. The following recommended films were received (Fig. 11):

	Назва фільму <chr>	Кількість елементів <fctr>
50	Usual Suspects, The (1995)	39
858	Godfather, The (1972)	34
318	Shawshank Redemption, The (1994)	32
527	Schindler's List (1993)	32

4 rows

Fig. 11. The most relevant movies according to UBCF algorithm

Comparing UBCF results with IBCF, you can find useful information about these algorithms. UBCF must have access to the initial data and keep the entire set of information in memory that often complicates work with a large matrix of ratings. In addition, building matrix of similarity requires a lot of computing power and time. However, UBCF accuracy is better than IBCF, and UBCF is a good choice if the data set is not very big [17]. The “VikToFilm” web application is a recommender system based on UBFC (User-based Collaborative Filtering) algorithm. The user interface was created the Shiny framework (Fig. 12). The server logic of the web application is described in the file "server.R", the logic of the user interface is described in the file "ui.R". As this web, application is cross-browser and multi-platform, it can be used from any device that has a browser and access into Internet (Fig. 13). You can access the application from any device with a browser and access to the Internet. The entrance points in the web application are its main page. The input data of program are a movie and rating database contained in movies.csv and ratings.csv. In order to develop a friendly web-application interface, the input data were simplified to select three genres and three films of the appropriate genres. The source data is a list of the ten most relevant movies for the user.

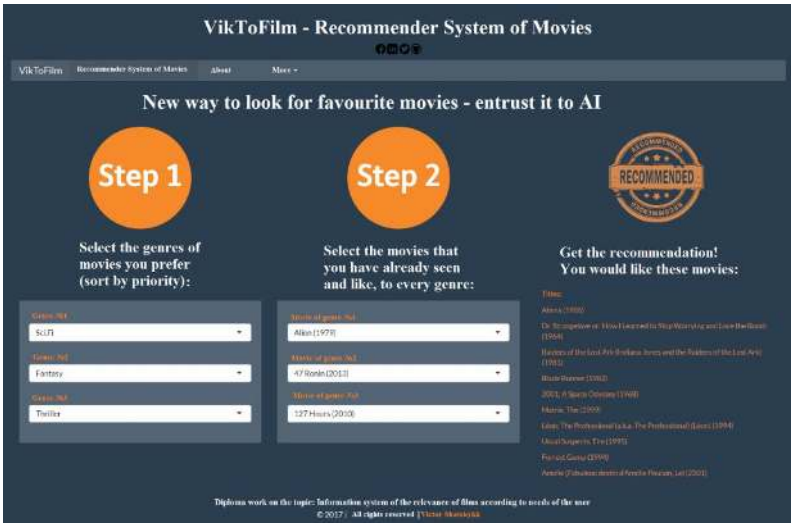

Fig. 12. Interface of main page of web application

Fig. 13. An example of web application working on an Android device

To verify the performance of the developed web application and accordance of system functioning results to task, the control example was conducted (Fig. 14-15).

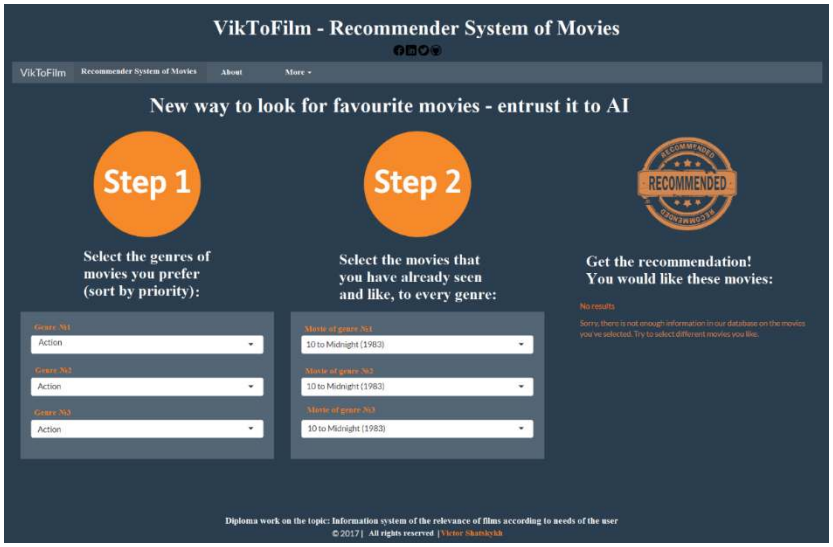

Fig. 14. Condition of the system at the beginning of work

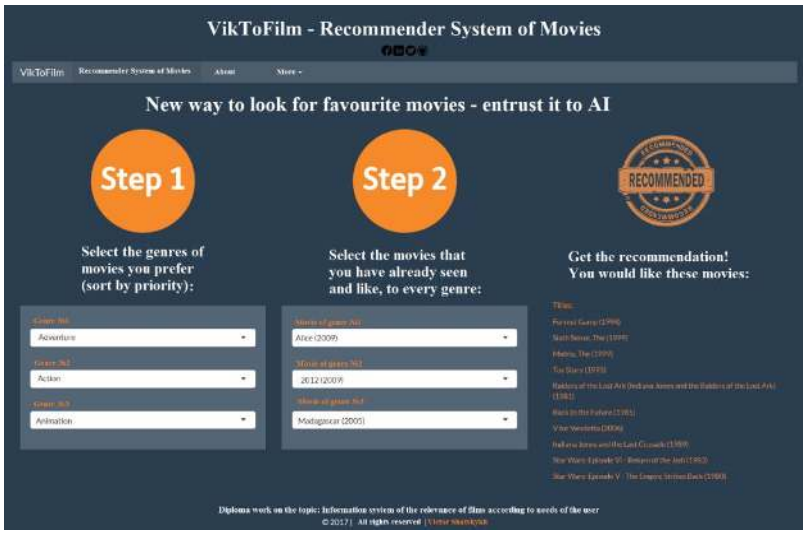

Fig. 15. Condition of the system at the end of work

As seen on the figures above (Fig. 14-15), the system works and provides rather relevant recommendations for selecting films.

5 User`s Manual

Web application "VikToFilm" is a recommender system of films based on UBCF (User-based Collaborative Filtering) algorithm. It is recommended to use the created system to collect information about the preferences of different people in the selection of films and provide relevant daily recommendations. The web app is designed to give you recommendations for choosing movies. Since the web application you created is cross-browser and multi-platform, you can use it from any browser with Internet access. The input data of program are a movie and rating database contained in movies.csv and ratings.csv. In order to develop a friendly web-application interface, the input data were simplified to select three genres and three films of the appropriate genres. The output data is a list of the ten most relevant movies for the user.

The web application can generate the following messages:

- «No results. Sorry, there is not enough information in our database about the movies you have selected. Try to select different movies you like. "- This means that the system lacks data to provide recommendations for selected movies, and you need to select other movies to get relevant recommendations.
- "Titles:" - list of names and years of screening of ten films, which, in the opinion of the system, are most relevant according to the input data.

During the development of system, the following environments and programming languages were used:

- programming language R version 3.4.0;
- JavaScript language;
- jQuery library;
- SockJS library;
- HTML5 / CSS3;
- Bootstrap framework version 3.3.6;
- Shiny version 1.0.2 framework;
- R-language development environment RStudio IDE version 1.0.136;
- Windows 10 Pro operating system.

It is recommended to use the created system to collect information about the preferences of different people in the selection of films and provide relevant daily recommendations. The program is able to provide relevant recommendations for choosing movies according to the needs and preferences of the user. The created system solves the tasks related to prediction for movie recommendations, the choice of movies that will be relevant to each individual user based on his preferences. In the software system for solving problems, the following methods are used:

- User-based Collaborative Filtering;
- Item-based Collaborative Filtering;
- Intellectual data analysis;
- Machine learning;
- Cosine coefficient;
- Pearson correlation coefficient;
- Jacquard coefficient.

The program offers relevant movie recommendations based on the selection of three feature-rich movie genres and three movie-related genres that he prefers.

Using this system, the user will greatly speed up his movie searches, which he would like to see. The system is hosted on cloud servers and operates around the clock. The Shiny service admin panel records the permanent logs of a web application that you can view at any time. If a web application uses too much traffic that can cause its slow work, then the service notifies the developer or administrator using email. The program can only be used to get movie recommendations.

To provide the stable work of web application, we recommend using a high quality and fast Internet connection, such as Wi-Fi or 3G, and a device with a sufficient amount of RAM. The application can be used from any device with a browser and access to the Internet. Peripherals are not required for work with the system. In order to expand the application on the Shiny Web servers, you need to have a locally installed language development environment R version 3.2 or higher, such as RStudio IDE. To install and use RStudio IDE version 1.0.136, any computer running Linux/Unix, Windows, or Mac OS X with at least 1 gigabytes of RAM and a 2 GHz processor can be used. You also need to have CRAN installed - an archive of network packets for the programming language R and an reconnect r package for connecting to Shiny servers and deploying them to the project.

6 Conclusion

During the performing of scientific work, an analysis of the subject area of the creation of movie recommender systems was implemented; approaches to the formation of relevant recommendations were analyzed. Comparison of existing recommender systems and methods of forming recommendations in such systems is executed. The methods of content filtering, collaborative filtration and hybrid methods are considered. The analysis of collaborative filtration algorithms is executed. The choice of collaborative filtering for the formulation of film recommendations for system users is substantiated. The expediency of developing an information system for the relevance of films to the needs of the user is substantiated. The choice of the programming R-language for implementation of the system is substantiated, the advantages and disadvantages of this language are described; the possibilities of its application, its main features are specified. The choice of the environment of RStudio IDE for the writing and debugging of the application code is substantiated. The technical specifications of the shinyapps.io server that was chosen to deploy the web application infrastructure are given. A practical implementation of the information recommender system for the selection of relevant films according to the needs of user based on the developed algorithm is executed. The work of the information system of the relevance of films to the needs of the user is analyzed. An algorithm of forming system recommendations based on collaborative filtration has been improved.

As a result of practical part of the work, the system was developed as a web-based application "VikToFilm" as a recommender system of films based on the UBCF (User-based Collaborative Filtering) algorithm. The description of the software, general information, functional purpose, description of the logical structure, used hardware, call and loading, input and output data of the system are given. An analysis of a control example of the program is given. The user's manual is described.

Economic analysis of the software product is executed. Strategies for developing a new product and its modification are selected. The strategy of developing a new product is characterized by the creation of completely new software, which allows you to solve the just created needs of people, society, economy, etc. The product development strategy is chosen because there are foreign analogues, but Ukrainian-language analogues have not been created yet. It is recommended that the system be used to collect information about the preferences of different people in the selection of films and provide appropriate relevant recommendations. In further studies in this area, it is expedient to improve the way to store movie data and replace the movie-ranking algorithm based on reviews, by based on a hybrid algorithm that takes into attention the qualitative and quantitative characteristics of reviews.

References

1. Melville, P., Mooney, R., Nagarajan, R.: Content-Boosted Collaborative Filtering for Improved Recommendations. National Conference on Artificial Intelligence: «AAAI-2002», 187–192. (2016).
2. Lytvyn, V., Vysotska, V., Demchuk, A., Demkiv, I., Ukhanska, O., Hladun, V. et. al.: Design of the architecture of an intelligent system for distributing commercial content in the

internet space based on SEO-technologies, neural networks, and Machine Learning. Eastern-European Journal of Enterprise Technologies, 2 (2 (98)), 15–34. (2019).
3. Jones, M. T.: Recommender systems, Part 1. Introduction to approaches and algorithms, https://www.ibm.com/developerworks/opensource/library/os-recommender1, last accessed 2019/07/21.
4. Su, X., Khoshgoftaar, T. M.: A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, 2009, 1–19. (2009).
5. Burov, Y., Vysotska, V., Kravets, P.: Ontological approach to plot analysis and modeling. Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Systems (COLINS-2019). Volume I: Main Conference, 2362, 22–31. (2019).
6. Sarwar, B., Karypis, G., Konstan, J., Reidl, J. Item-based collaborative filtering recommendation algorithms. Proceedings of the Tenth International Conference on World Wide Web - WWW '01. (2001).
7. Schafer, J. B., Konstan, J., Riedi, J.: Recommender systems in e-commerce. Proceedings of the 1st ACM Conference on Electronic Commerce - EC '99. (1999).
8. Gope, J., Jain, S. K.: A survey on solving cold start problem in recommender systems. 2017 International Conference on Computing, Communication and Automation (ICCCA). (2017).
9. Ge, M., Delgado-Battenfeld, C., Jannach, D.: Beyond accuracy: Evaluating recommender systems by coverage and serendipity. Proceedings of the fourth ACM conference on Recommender systems - RecSys '10, 257–260. (2010).
10. Bobadilla, J., Ortega, F., Hernando, A., Bernal, J.: A collaborative filtering approach to mitigate the new user cold start problem. Knowledge-Based Systems, 26, 225–238. (2012).
11. Nambiar, R., Bhardwaj, R., Sethi, A., Vargheese, R.: A look at challenges and opportunities of Big Data analytics in healthcare. International Conference on Big Data. (2013).
12. Calero Valdez, A., Ziefle, M., Verbert, K.: HCI for recommender systems: The past, the present and the future. RecSys '16 Proceedings of the 10th ACM Conference on Recommender System, 123–126. (2016).
13. Kotsiantis, S. B., Zaharakis, I., Pintelas, P.: Supervised machine learning: A review of classification techniques. Frontiers in Artificial Intelligence and Applications, Vol. 160: Emerging Artificial Intelligence Applications in Computer Engineering, 3–24. (2007).
14. Recommended For You FAQ, https://help.imdb.com/article/imdb/discover-watch/recommended-for-you-faq/GPZ2RSPB3CPVL86Z/, last accessed 2019/07/21.
15. Netflix Prize,: https://www.netflixprize.com/, last accessed 2019/07/21.
16. About Rotten Tomatoes, https://www.rottentomatoes.com/about, last accessed 2019/07/21.
17. Lytvyn, V., Vysotska, V., Rzheuskyi, A.: Technology for the Psychological Portraits Formation of Social Networks Users for the IT Specialists Recruitment Based on Big Five, NLP and Big Data Analysis. Proceedings of the 1st International Workshop on Control, Optimisation and Analytical Processing of Social Networks (COAPSN-2019), 2392, 147–171. (2019).
18. Lytvyn, V., Vysotska, V., Rusyn, B., Pohreliuk, L., Berezin, P., Naum, O.: Textual Content Categorizing Technology Development Based on Ontology. Workshop Proceedings of the 8th International Conference on "Mathematics. Information Technologies. Education", 2386, 234–254. (2019).
19. Lytvyn, V., Kuchkovskiy, V., Vysotska, V., Markiv, O., Pabyrivskyy, V.: Architecture of System for Content Integration and Formation Based on Cryptographic Consumer Needs. 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT). (2018).
20. Rubens, N., Elahi, M., Sugiyama, M., Kaplan, D.: Active Learning in Recommender Systems. Recommender Systems Handbook, 809–846. (2015).

21. Ms. Ashwini A. Chirde, Ms. Urmila K.: Combination of a Cluster-Based and Content-Based Collaborative Filtering Approach for Recommender System. In: Recent and Innovation Trends in Computing and Communication, 3 (7), 4770–4774. (2015).
22. Harper, F. M., Konstan, J. A.: The MovieLens Datasets. ACM Transactions on Interactive Intelligent Systems, 5 (4), 1–19. (2015).
23. Grolemund, G.: Hands-On Programming with R: Write Your Own Functions and Simulations. Sebastopol, United States. (2015).
24. McLeod, D., Chen, A.-Y.: Collaborative Filtering for Information Recommendation Systems. Research Reports. (2009).
25. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook. Springer. (2015).
26. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7 (1), 76–80. (2003).
27. The Comprehensive R Archive Network, https://cran.r-project.org, last accessed 2019/07/21.
28. RStudio, https://www.rstudio.com/products, last accessed 2019/07/21.
29. Chapter 2 Getting Started, https://docs.rstudio.com/shinyapps.io/getting-started.html, last accessed 2019/07/21.
30. MovieLens Latest Datasets, https://grouplens.org/datasets/movielens/latest, last accessed 2019/07/21.
31. Sitecore Documentation: Access all the latest Sitecore documentation, https://doc.sitecore.com, last accessed 2019/07/21.
32. Nouh, R., Lee, H.-H., Lee, W.-J., Lee, J.-D.: A Smart Recommender Based on Hybrid Learning Methods for Personal Well-Being Services. Sensors, 19 (2), 431. (2019).
33. Mobasher, B.: Data Mining for Web Personalization. Lecture Notes in Computer Science, 90–135. (2007).
34. Berko, A., Alieksieiev, V.: A Method to Solve Uncertainty Problem for Big Data Sources. 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). (2018).
35. Xu, G., Zhang, Y., Li, L.: Web Content Mining. Web Mining and Social Networking, 71–87. (2010).
36. Lytvyn, V., Vysotska, V., Pukach, P., Nytrebych, Z., Demkiv, I., Senyk, A. et. al.: Analysis of the developed quantitative method for automatic attribution of scientific and technical text content written in Ukrainian. Eastern-European Journal of Enterprise Technologies, 6 (2 (96)), 19–31. (2018).
37. Gozhyj, A., Kalinina, I., Vysotska, V., Gozhyj, V.: The Method of Web-Resources Management Under Conditions of Uncertainty Based on Fuzzy Logic. 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT). (2018).
38. Lytvyn, V., Vysotska, V., Dosyn, D., Burov,Y.: Method for ontology content and structure optimization, provided by a weighted conceptual graph. Webology, 15(2), 66–85. (2018).
39. Khomytska, I., Teslyuk, V.: Specifics of phonostatistical structure of the scientific style in English style system. 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT). (2016).
40. Khomytska, I., Teslyuk, V.: The Method of Statistical Analysis of the Scientific, Colloquial, Belles-Lettres and Newspaper Styles on the Phonological Level. Advances in Intelligent Systems and Computing, 149–163. (2016).
41. Nytrebych, Z. M., Malanchuk, O. M., Il'kiv, V. S., Pukach, P. Ya.: Homogeneous problem with two-point conditions in time for some equations of mathematical physics. Azerbaijan Journal of Mathematics, 7 (2), 180–196. (2017).
42. Nytrebych, Z., Il'kiv, V., Pukach, P., Malanchuk, O.: On nontrivial solutions of homogeneous Dirichlet problem for partial differential equations in a layer. Kragujevac Journal of Mathematics, 42 (2), 193–207. (2018).

43. Nytrebych, Z., Malanchuk, O., Il'kiv, V., Pukach, P.: On the solvability of two-point in time problem for PDE. Italian Journal of Pure and Applied Mathematics, 38, 715–726. (2017).
44. Pukach, P. Ya., Kuzio, I. V., Nytrebych, Z. M., Ilkiv, V. S.: Analytical methods for determining the effect of the dynamic process on the nonlinear flexural vibrations and the strength of compressed shaft. Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu, 5, 69–76. (2017).
45. Pukach, P. Y., Kuzio, I. V., Nytrebych, Z. M., Il'kiv, V. S.: Asymptotic method for investigating resonant regimes of nonlinear bending vibrations of elastic shaft. Scientific Bulletin of National Mining University, 1, 68–73. (2018).
46. Nytrebych, Z., Ilkiv, V., Pukach, P., Malanchuk, O., Kohut, I., Senyk, A.: Analytical method to study a mathematical model of wave processes under two-point time conditions. Eastern-European Journal of Enterprise Technologies, 1 (7 (97)), 74–83. (2019).
47. Pukach, P., Il'kiv, V., Nytrebych, Z., Vovk, M., Pukach, P.: On the Asymptotic Methods of the Mathematical Models of Strongly Nonlinear Physical Systems. Advances in Intelligent Systems and Computing, 421–433. (2018).
48. Lavrenyuk, S. P., Pukach, P. Y.: Mixed problem for a nonlinear hyperbolic equation in a domain unbounded with respect to space variables. Ukrainian Mathematical Journal, 59 (11), 1708–1718. (2007).
49. Pukach, P. Y.: Investigation of Bending Vibrations in Voigt–Kelvin Bars with Regard for Nonlinear Resistance Forces. Journal of Mathematical Sciences, 215 (1), 71–78. (2016).
50. Pukach, P., Il'kiv, V., Nytrebych, Z., Vovk, M.: On nonexistence of global in time solution for a mixed problem for a nonlinear evolution equation with memory generalizing the Voigt-Kelvin rheological model. Opuscula Mathematica, 37 (45), 735. (2017).
51. Pukach, P. Y.: On the unboundedness of a solution of the mixed problem for a nonlinear evolution equation at a finite time. Nonlinear Oscillations, 14 (3), 369–378. (2012).
52. Pukach, P. Y.: Qualitative Methods for the Investigation of a Mathematical Model of Nonlinear Vibrations of a Conveyer Belt. Mathematical Sciences, 198, 31–38. (2014).
53. Bezobrazov, S., Sachenko, A., Komar, M., Rubanau, V.: The Methods of Artificial Intelligence for Malicious Applications Detection in Android OS. International Journal of Computing, 15 (3), 184–190. (2016).
54. Mashkov V.A., Barabash O. V.: Self-checking and self-diagnosis of module systems on the principle of walking diagnosis kernel. In: Engineering Simulation, 15, 43-51. (1998)
55. Dunets, O., Wolff, C., Sachenko, A., Hladiy, G., Dobrotvor, I.: Multi-agent system of IT project planning. 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). (2017).
56. Lytvyn, V., Vysotska, V., Pukach, P., Nytrebych, Z., Demkiv, I., Kovalchuk, R., Huzyk, N.: Development of the linguometric method for automatic identification of the author of text content based on statistical analysis of language diversity coefficients. Eastern-European Journal of Enterprise Technologies, 5 (2 (95)), 16–28. (2018).
57. Vysotska, V., Lytvyn, V., Burov, Y., Berezin, P., Emmerich, M., Basto Fernandes, V.: Development of Information System for Textual Content Categorizing Based on Ontology. CEUR Workshop Proceedings, 53–70. (2019).
58. Vysotska, V., Lytvyn, V., Burov, Y., Gozhyj, A., Makara, S.: The consolidated information web-resource about pharmacy networks in city. CEUR Workshop Proceedings, 2255, 239–255. (2018).
59. Rusyn, B., Vysotska, V., Pohreliuk, L.: Model and Architecture for Virtual Library Information System. Computer Sciences and Information Technologies (CSIT). (2018).
60. Lytvyn, V., Vysotska, V., Dosyn, D., Lozynska, O., Oborska, O.: Methods of Building Intelligent Decision Support Systems Based on Adaptive Ontology. 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). (2018).

61. Lytvyn, V., Vysotska, V., Burov, Y., Bobyk, I., Ohirko, O.: The Linguometric Approach for Co-authoring Author's Style Definition. 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS). (2018).
62. Demchuk, A., Lytvyn, V., Vysotska, V., Dilai, M.: Methods and Means of Web Content Personalization for Commercial Information Products Distribution. Lecture Notes in Computational Intelligence and Decision Making, 332–347. (2020).
63. Veres, O., Rusyn, B., Sachenko, A., Rishnyak, I.: Choosing the method of finding similar images in the reverse search system. CEUR Workshop Proceedings, 2136, 99–107. (2018).
64. Babichev, S., Durnyak, B., Pikh, I., Senkivskyy, V.: An Evaluation of the Objective Clustering Inductive Technology Effectiveness Implemented Using Density-Based and Agglomerative Hierarchical Clustering Algorithms. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 532-553. (2020)
65. Vysotska, V., Lytvyn, V., Hrendus, M., Kubinska, S., Brodyak, O.: Method of Textual Information Authorship Analysis Based on Stylometry. International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT). (2018).
66. Bisikalo, O., Ivanov, Y., Sholota, V.: Modeling the Phenomenological Concepts for Figurative Processing of Natural-Language Constructions. In: CEUR Workshop Proceedings, Vol- 2362, 1-11. (2019)
67. Vysotska, V., Burov, Y., Lytvyn, V., Demchuk, A.: Defining Author's Style for Plagiarism Detection in Academic Environment. 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), 128–133. (2018).
68. Babichev, S., Gozhyj, A., Kornelyuk A., Litvinenko, V.: Objective clustering inductive technology of gene expression profiles based on SOTA clustering algorithm. In: Biopolymers and Cell, 33(5), 379–392. (2017)
69. Babichev, S., Taif, M.A., Lytvynenko, V., Osypenko, V.: Criterial analysis of gene expression sequences to create the objective clustering inductive technology. In: 2017 IEEE 37th International Conference on Electronics and Nanotechnology, 244-248. (2017)
70. Lypak, H.,.Rzheuskyi, A., Kunanets, N., Pasichnyk, V: Formation of a consolidated information resource by means of cloud technologies. In: International Scientific-Practical Conference on Problems of Infocommunications Science and Technology. (2018)
71. Rzheuskyi, A., Kunanets, N., Stakhiv, M.: Recommendation System: Virtual Reference. In: 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), 203-206. (2018)
72. Kaminskyi, R., Kunanets, N., Rzheuskyi, A.: Mathematical support for statistical research based on informational technologies. CEUR Workshop Proceedings,2105, 449-452 (2018).
73. Kazarian, A., Kunanets, N., Pasichnyk, V., Veretennikova, N., Rzheuskyi, A., Leheza, A., Kunanets, O.: Complex Information E-Science System Architecture based on Cloud Computing Model. In: CEUR Workshop Proceedings, Vol- 2362, 366-377. (2019)
74. Veres, O., Rishnyak, I., Rishniak, H.: Application of Methods of Machine Learning for the Recognition of Mathematical Expressions. In: CEUR Workshop Proceedings, Vol- 2362, 378-389. (2019)
75. Kravets, P.: The control agent with fuzzy logic. Perspective Technologies and Methods in MEMS Design, 40–41. (2010).
76. Kravets, P.: The Game Method for Orthonormal Systems Construction. The Experience of Designing and Applications of CAD Systems in Microelectronics. (2007).
77. Kravets, P.: Game Model of Dragonfly Animat Self-Learning. Perspective Technologies and Methods in MEMS Design, 195–201. (2016).
78. Sachenkom, S., Lendyuk, T., Rippa, S.: Simulation of Computer Adaptive Learning and Improved Algorithm of Pyramidal Testing. In: International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2, 764-770. (2013)

79. Sachenko, S., Lendyuk, T., Rippa, S., Sapojnyk, G.: Fuzzy Rules for Tests Complexity Changing for Individual Learning Path Construction. In: Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 945-948. (2015)
80. Teslyuk, V., Beregovskyi, V., Denysyuk, P., Teslyuk, T., Lozynskyi, A.: Development and Implementation of the Technical Accident Prevention Subsystem for the Smart Home System. International Journal of Intelligent Systems and Applications, 10 (1), 1–8. (2018).
81. Basyuk, T.: The main reasons of attendance falling of internet resource. Computer Sciences and Information Technologies (CSIT). (2015).
82. Chernukha, O., Bilushchak, Y.: Mathematical modeling of random concentration field and its second moments in a semispace with erlangian disrtibution of layered inclusions. Task Quarterly, 20 (3), 295–334. (2016).
83. Rzheuskyi, A., Gozhyj, A., Stefanchuk, A., Oborska, O., Chyrun, L., Lozynska, O., Mykich, K., Basyuk, T.: Development of Mobile Application for Choreographic Productions Creation and Visualization. In: CEUR Workshop Proceedings, 2386, 340-358. (2019)
84. Mukalov, P., Zelinskyi, O., Levkovych, R., Tarnavskyi, P., Pylyp, A., Shakhovska, N.: Development of System for Auto-Tagging Articles, Based on Neural Network. In: CEUR Workshop Proceedings, Vol-2362, 106-115. (2019)
85. Romanenkov, Y. Pasichnyk, V., Veretennikova, N., Nazaruk, M., Leheza, A.: Information and Technological Support for the Processes of Prognostic Modeling of Regional Labor Markets. In: CEUR Workshop Proceedings, Vol-2386, 24-34. (2019)
86. Levchenko, O., Romanyshyn, N., Dosyn, D.: Method of Automated Identification of Metaphoric Meaning in Adjective + Noun Word Combinations (Based on the Ukrainian Language). In: CEUR Workshop Proceedings, Vol-2386, 370-380. (2019)
87. Mashkov, V.A., Barabash O. V.: Self-checking and self-diagnosis of module systems on the principle of walking diagnosis kernel. // Engineering Simulation, Vol.15, 43-51. (1998)
88. Lytvynenko, V., Savina, N., Krejci, J., Voronenko, M., Yakobchuk, M., Kryvoruchko, O.: Bayesian Networks' Development Based on Noisy-MAX Nodes for Modeling Investment Processes in Transport. CEUR Workshop Proceedings, 2386, 1–10. (2019).
89. Lytvynenko, V., Lurie, I., Krejci, J., Voronenko, M., Savina, N., Taif, M. A.: Two Step Density-Based Object-Inductive Clustering Algorithm. CEUR Workshop Proceedings, 117–135. (2019).