FITPED | AI

Recommender Systems

Radim Farana
Cyril Klimeš
Ján Skalka

www.fitped.eu

2024

Funded by the European Union

Erasmus+ FITPED-AI
Future IT Professionals Education in Artificial Intelligence
(Project 2021-1-SK01-KA220-HED-000032095)

FITPED | AI

Recommender Systems

Published on

November 2024

Authors

Radim Farana | Mendel University in Brno, Czech Republic

Cyril Klimeš | Mendel University in Brno, Czech Republic

Ján Skalka | Constantine the Philosopher University in Nitra, Slovakia

Reviewers

Piet Kommers | Helix5, Netherland

Małgorzata Przybyła-Kasperek | University of Silesia in Katowice, Poland

Vladimiras Dolgopolovas | Vilnius University, Lithuania

Erasmus+ FITPED-AI

Future IT Professionals Education in Artificial Intelligence

Project 2021-1-SK01-KA220-HED-000032095

Funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Slovak Academic Association for International Cooperation. Neither the European Union nor the granting authority can be held responsible for them.

ISBN 978-80-558-2236-5

TABLE OF CONTENTS

Introduction

Chapter 1

1.1 Definition

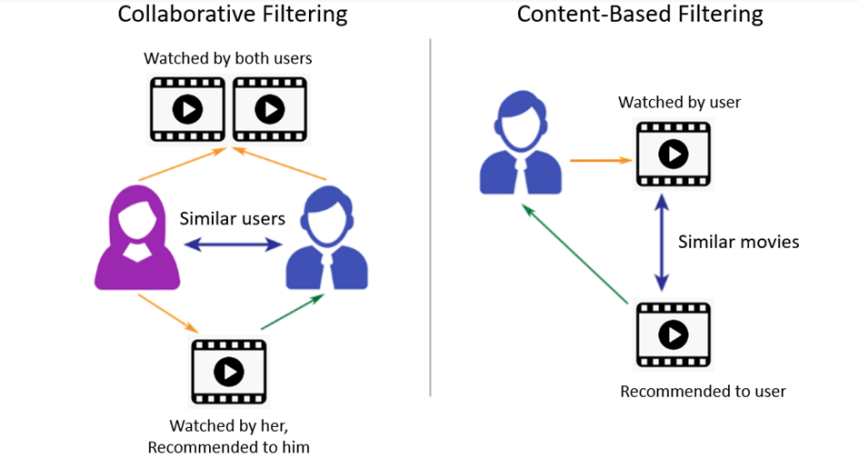
🕮 1.1.1

Recommender systems, also known as recommendation systems, are specialized software applications that leverage artificial intelligence to make tailored suggestions for users. According to [Wikipedia], a recommender system is a "a subclass of information filtering system that provide suggestions for items that are most pertinent to a particular user. Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what online news to read. Recommender systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer.“

These systems assist users in various decision-making processes, such as selecting a product, choosing music, or picking news articles to read online. Recommender systems are particularly valuable when users face an overwhelming number of options, enabling them to focus on relevant choices rather than sifting through countless items. As outlined in [Mining of Massive Datasets. Chapter 9 Recommendation systems] these systems can broadly be classified into two types:

- **Content-based filtering** focuses on analyzing the features of items a user has already expressed interest in. For example, if a Netflix user has previously watched cowboy movies, the system might suggest more movies within the "cowboy" genre. This type of system is highly personalized, relying on the user’s individual viewing history and preferences to drive recommendations.
- **Collaborative filtering**, on the other hand, identifies similarities between users and/or items to make suggestions. In this approach, items recommended to a user are often those that similar users have preferred. This technique leverages similarity measures and can utilize clustering techniques to find patterns among users’ preferences. However, while content-based and collaborative filtering are foundational methods in recommender systems, they aren’t always sufficient on their own. New algorithms and hybrid models are continuously being developed to improve recommendation accuracy and effectiveness, allowing recommender systems to adapt to users' evolving preferences more effectively.

The main idea of both systems is present in the picture:



1.1.2

Which method in recommender systems analyzes item features that a user has shown interest in?

- Content-based filtering
- Collaborative filtering
- Hybrid filtering
- Pattern-based filtering

1.1.3

Select the correct statements about recommender systems.

- Content-based filtering focuses on item properties.
- Collaborative filtering identifies similar users' preferences.
- Recommender systems help in decision-making by providing relevant suggestions.
- Collaborative filtering only uses individual user preferences.

1.1.4

Content-based filtering

Content-based filtering works by identifying items with similar content, focusing on the user's interests to recommend related items. When a user watches a movie, the system analyzes its attributes—such as genre, director, or keywords associated with

its content. It then suggests other movies with matching or similar characteristics. This is different from collaborative filtering, where recommendations are based on preferences of other users with similar tastes. Content-based filtering doesn't rely on other users' data; instead, it requires a description of the item itself, which makes it highly personalized for each user.

Unlike collaborative filtering, which leverages group data, content-based filtering utilizes additional details about the user and items to make individualized predictions. For instance, descriptive keywords from a movie, like "thriller," "action," or "historical," are compared with past selections by the user to highlight preferences. This method enables more refined recommendations by associating a user's viewing history with content attributes. It means that the only history necessary for making predictions is that of the target user, which can be highly effective, especially when group preferences aren't available or relevant.

Content-based filtering also uses feature extraction techniques to analyze descriptive aspects of the items. By breaking down a movie's description into keywords, genres, or other meaningful attributes, the system can match these features with the user's past choices. This personalized approach offers precise recommendations, especially when the user's preferences are clearly reflected in the item's content. Because content-based filtering does not depend on others' ratings, it works well when items or users lack extensive rating data, ensuring a tailored recommendation experience based on each user's unique interests.

1.1.5

Content-based filtering focuses on _____ similar content to recommend _____ items to the user, unlike _____ filtering, which relies on other users' preferences.

- analyzing
- similar
- collaborative

1.1.6

Collaborative filtering

Collaborative filtering systems are among the most widely used and advanced recommendation technologies on the market. The defining concept of collaborative filtering is its "collaborative" nature, meaning it relies on evaluations and ratings from other users. For example, if you want to know whether you'd enjoy a particular movie, you could ask people with similar tastes for their opinions. By gathering input from users with similar preferences, the system can build a list of recommendations based on their ratings and past interactions. Collaborative filtering gathers user ratings on

items, identifies shared characteristics among users based on these ratings, and generates new recommendations by comparing user preferences.

Collaborative filtering methods can be divided into two main categories:

- **Memory-based** methods, often referred to as neighborhood-based approaches, leverage other users' ratings to predict how a target user might rate an item. This can be done through two variants: user-based, where similar users are identified, and item-based, where similar items are used to recommend based on the user's preferences. These methods rely on historical data without creating complex models, making them relatively straightforward yet effective when sufficient user data is available.
- **Model-based** collaborative filtering takes the technique a step further, incorporating advanced machine learning models such as decision trees, latent factor models, and neural networks to make predictions. Instead of directly using raw ratings, model-based methods use statistical techniques to uncover underlying patterns and trends in user behavior, allowing for more complex and nuanced recommendations.

1.1.7

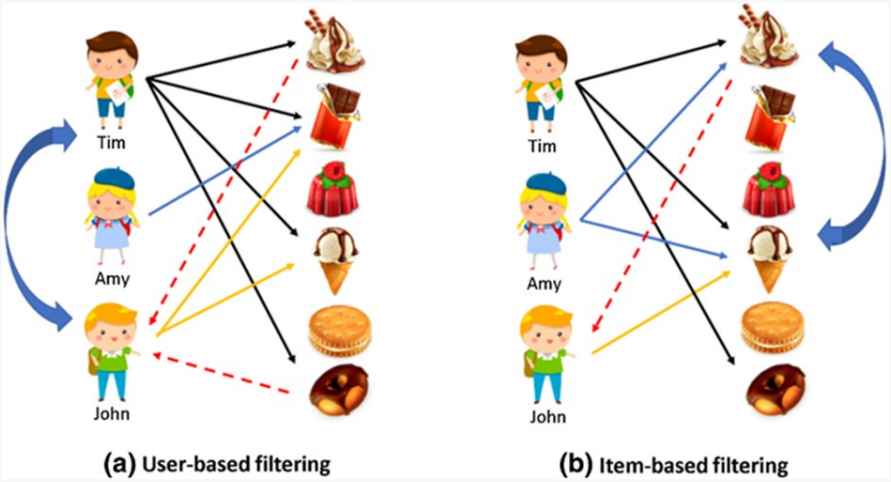
Collaborative filtering relies on _____ from other users, recognizing shared preferences among users. Memory-based methods focus on _____ ratings, while model-based methods use _____ learning for recommendations.

- analyzing
- machine
- evaluations

1.1.8

Memory-based or **neighborhood-based** system collects behavioral information about how you interact with items - what ratings you have, how you rate, what you view and/or what you purchase. One typical approach is to create a user rating matrix containing user ratings on movies. We find similarities and make recommendations. Similarity is not only limited to the user's taste, in addition, similarity between different items can also be considered. If we have a large amount of information about users and items, the system will provide more effective recommendations.

Example:



In the collaborative filtering above, there are three users Tim, Amy and John who are interested in desserts. The system detects users who have the same taste in purchasing products, and the similarity between users is calculated based on purchasing behavior.

- **User-based collaborative filtering**: Tim and John are similar because they bought similar products.
- **Item-based collaborative filtering**: the system checks for items similar to items the user has purchased. The similarity between different items is calculated based on items and not users for prediction. Tim and Amy bought an ice cream sundae and an ice cream cone, so it turns out they have similar tastes.

The application domain for recommender systems is very wide. Typical application are known form movie databases, which we will also use. But they are many applications for business purposes [Recommender Systems For Business - A Gentle Introduction] available. Many possibilities for recommender systems are still waiting for solution.

1.1.9

Which collaborative filtering method uses machine learning models to predict user preferences?

- Model-based
- Neighborhood-based
- Memory-based
- Item-based

1.1.10

Which filtering method uses only the user's preferences to recommend items?

- Content-based filtering
- Collaborative filtering
- Social filtering
- Collective filtering

1.1.11

The main methods used in recommender systems are:

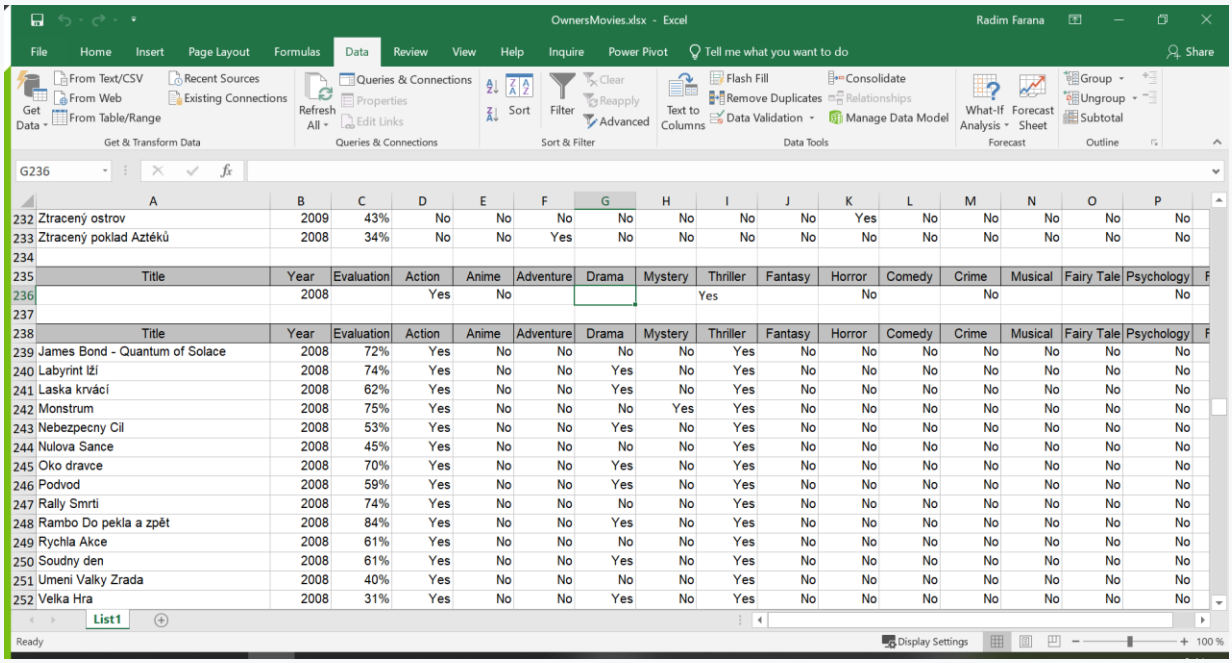
- Memory-based methods
- Model-based methods
- Matrix-based methods
- Moutain-climbing methods

1.2 Easy example – movie selection

1.2.1

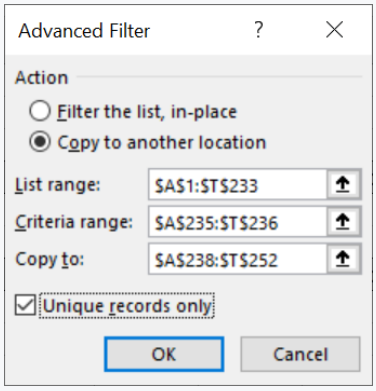
Typical recommender systems applications are known from the movie industry. Movie portals offers so many movies that it is difficult to navigate them and it is very difficult to choose a suitable movie for a specific occasion. To facilitate the choice, we can define a set of parameters (for example “Drama”, “Sci-fi”, etc.) and assign them to all movies.

The selection problem could be presented by the use of some owner’s movie database in MS-Excel (that is why the names of movies are in owner’s language, file is for your experiments available here: https://priscilla.fitped.eu/data/recommender_systems/OwnersMovies.xlsx), which allow us to select appropriate movies with the help of more parameters:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
232	Ztracený ostrov	2009	43%	No	No	No	No	No	No	No	Yes	No	No	No	No	No
233	Ztracený poklad Aztéků	2008	34%	No	No	Yes	No	No	No	No	No	No	No	No	No	No
234																
235	Title	Year	Evaluation	Action	Anime	Adventure	Drama	Mystery	Thriller	Fantasy	Horror	Comedy	Crime	Musical	Fairy Tale	Psychology
236		2008		Yes	No				Yes		No		No			No
237																
238	Title	Year	Evaluation	Action	Anime	Adventure	Drama	Mystery	Thriller	Fantasy	Horror	Comedy	Crime	Musical	Fairy Tale	Psychology
239	James Bond - Quantum of Solace	2008	72%	Yes	No	No	No	No	Yes	No	No	No	No	No	No	No
240	Labyrint lží	2008	74%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
241	Laska krvácí	2008	62%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
242	Monstrum	2008	75%	Yes	No	No	No	Yes	Yes	No	No	No	No	No	No	No
243	Nebezpecny Cil	2008	53%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
244	Nulova Sance	2008	45%	Yes	No	No	No	No	Yes	No	No	No	No	No	No	No
245	Oko dravce	2008	70%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
246	Podvod	2008	59%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
247	Rally Smrti	2008	74%	Yes	No	No	No	No	Yes	No	No	No	No	No	No	No
248	Rambo Do pekla a zpět	2008	84%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
249	Rychla Akce	2008	61%	Yes	No	No	No	No	Yes	No	No	No	No	No	No	No
250	Soudny den	2008	61%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No
251	Umeni Valky Zrada	2008	40%	Yes	No	No	No	No	Yes	No	No	No	No	No	No	No
252	Velka Hra	2008	31%	Yes	No	No	Yes	No	Yes	No	No	No	No	No	No	No

For movie section based on selected parameters we can use the standard tool "Advanced filter":



Even this kombination of parametrs (Action Thriller which is not Anime, Horror, Crime nor Psychology) gave us some suggested results.

1.2.2

How many results do we get when we combine the parameters: action thriller that is not anime, horror, crime fiction or psychology?

1.2.3

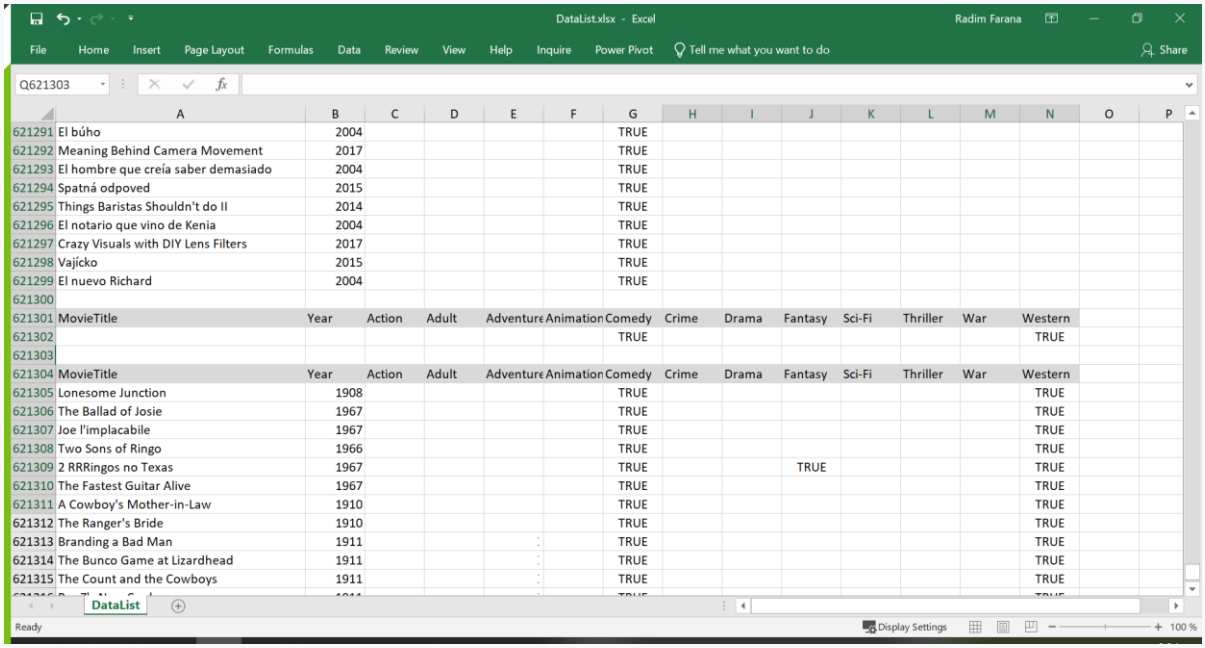
Which kind of filtering are we used in the example of movie selection, based on parameters in the file https://priscilla.fitped.eu/data/recommender_systems/OwnersMovies.xlsx?

- Content-based filtering
- Collaborative filtering

1.2.4

Even in this short list of movies we must use more parameters to obtain short list of possible movies. To understand how complicate problem is to recommend appropriate list of movies you can use for non-commercial purposes movie database available on web: https://datasets.imdbws.com/, documentation for these data files can be found on http://www.imdb.com/interfaces/ (encoding Unicode (UTF-8)).

Having so long list of movies it is practically impossible to obtain some short list of movies (the original file for your expertiments is available here: https://priscilla.fitped.eu/data/recommender_systems/DataList.xlsx), for example:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
621291	El búho	2004					TRUE									
621292	Meaning Behind Camera Movement	2017					TRUE									
621293	El hombre que creía saber demasiado	2004					TRUE									
621294	Spatná odpoved	2015					TRUE									
621295	Things Baristas Shouldn't do II	2014					TRUE									
621296	El notario que vino de Kenia	2004					TRUE									
621297	Crazy Visuals with DIY Lens Filters	2017					TRUE									
621298	Vajícko	2015					TRUE									
621299	El nuevo Richard	2004					TRUE									
621300																
621301	MovieTitle	Year	Action	Adult	Adventure	Animation	Comedy	Crime	Drama	Fantasy	Sci-Fi	Thriller	War	Western		
621302							TRUE							TRUE		
621303																
621304	MovieTitle	Year	Action	Adult	Adventure	Animation	Comedy	Crime	Drama	Fantasy	Sci-Fi	Thriller	War	Western		
621305	Lonesome Junction	1908					TRUE							TRUE		
621306	The Ballad of Josie	1967					TRUE							TRUE		
621307	Joe l'implacabile	1967					TRUE							TRUE		
621308	Two Sons of Ringo	1966					TRUE							TRUE		
621309	2 RRRingos no Texas	1967					TRUE			TRUE				TRUE		
621310	The Fastest Guitar Alive	1967					TRUE							TRUE		
621311	A Cowboy's Mother-in-Law	1910					TRUE							TRUE		
621312	The Ranger's Bride	1910					TRUE							TRUE		
621313	Branding a Bad Man	1911					TRUE							TRUE		
621314	The Bunco Game at Lizardhead	1911					TRUE							TRUE		
621315	The Count and the Cowboys	1911					TRUE							TRUE		

1.2.5

When using the list of available movies: https://priscilla.fitped.eu/data/recommender_systems/DataList.xlsx, how complicated is it to obtain a short list of recommended movies, based on the parameters selection?

- It is a piece of cake.
- It is practically impossible

Traditional Recommender Systems

Chapter 2

2.1 Collaborative filtering

🕮 2.1.1

A set of items that are dependent on the user's previous selections is necessary for collaborative filtering. For this system to function, not many product features are necessary. Each item and user are described by an embedding or feature vector, which sinks them both in the same embedding place. It independently generates enclosures for both users and objects.

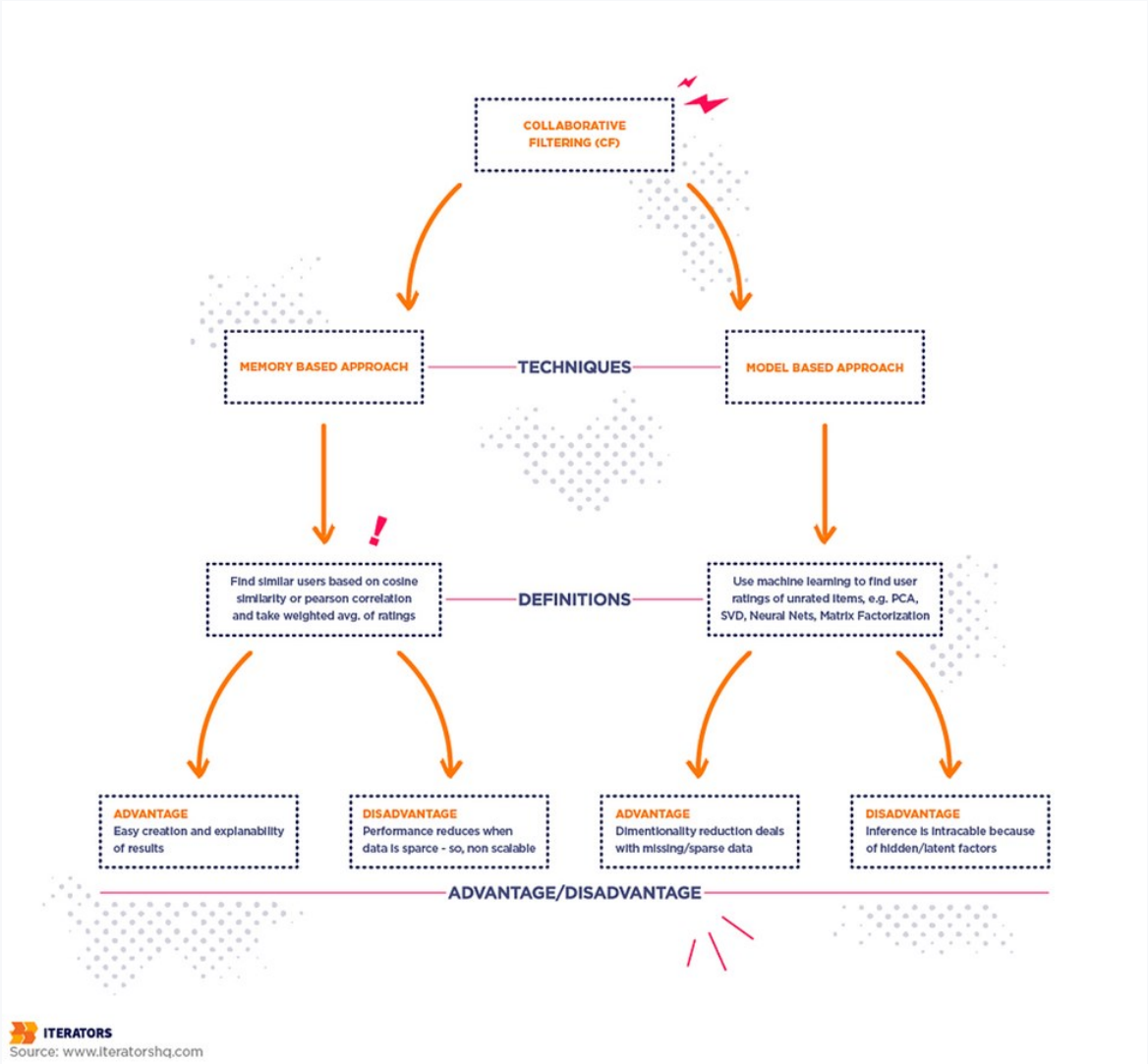
When recommending a specific product to the main user, the responses of other buyers are considered. Before proposing the item that customers like the most, it takes note of how each user behaves. When recommending a product to the main consumer, it also connects similar users based on similarities in preferences and behavior towards a related product.

The most well-known application suggestion engine, Collaborative Filtering, makes intelligent predictions about which users would like a given product in the future. This type of algorithm is also known as a product-based collaborative shift. Instead of things, in this filtering, users are screened and linked to each User. Only users' behavior is considered in this system. Only their profile information and content are insufficient. Users who rate products favorably will be linked to other users who act similarly favorably.

Types of Collaborative Filtering

There are two basic types of the collaborative filtering process:

1. Memory-based collaborative filtering
2. Model-based collaborative filtering



Collaborative Filtering (Source: www.iteratorshq.com)

2.1.2

What does collaborative filtering primarily rely on for making recommendations?

- Similarities in user preferences
- Product attributes
- User behavior
- Item descriptions

2.1.3

Which statements are true about collaborative filtering?

- It places users and items in the same vector space.
- It connects users with similar preferences.
- It requires detailed product features for accurate recommendations.
- It depends solely on profile information and content.

2.1.4

Memory-based collaborative filtering

Collaborative filtering (CF) is a technique used by recommender systems. Collaborative filtering has two senses, a narrow one and a more general one.

Collaborative filtering, in its more recent, condensed form, is a technique for making predictions (or filtering) about a user's interests automatically while simultaneously gathering preferences or taste data from many users. The collaborative filtering method's core premise is that if two people have the same view on a subject, they are more likely to have it on a different subject than two randomly selected people. For instance, given a partial list of a user's preferences (likes or dislikes), a collaborative filtering recommendation system for television programs could make predictions about which television show the user should like. It should be noted that although these forecasts are based on data from numerous users, they are specific to the individual.

This is distinct from the more straightforward method of assigning an average (non-specific) score for each interesting item, for instance, based on the number of votes it has received.

In a broader sense, collaborative filtering refers to searching for patterns or information by working with various agents, viewpoints, data sources, etc. Collaborative filtering applications sometimes include enormous data sets. Numerous types of data have been subjected to collaborative filtering techniques, such as sensing and monitoring data used in mineral exploration, environmental sensing over wide areas, or data collected by numerous sensors; financial data used by financial service providers who integrate numerous financial sources; or data used in e-commerce and web applications where the focus is on user data, etc. The rest of the discussion will concentrate on collaborative filtering for user data, however, some of the techniques and strategies may also apply to the other principal applications.

2.1.5

What is the core premise of collaborative filtering in its more specific sense?

- Predicting a user’s preferences based on similar users' preferences
- Assigning a general score based on item popularity
- Calculating the average rating for each item
- Using user demographics to make recommendations

2.1.6

Select the correct statements about collaborative filtering in a broader sense.

- It involves working with large data sets.
- It can analyze data from multiple sources like sensors and financial services.
- It exclusively focuses on user preferences for movies and TV shows.
- It only applies to e-commerce and web applications.

2.1.7

Model-based collaborative filtering

Model-based collaborative filtering provides recommendations by developing a model from user ratings. In addition to using explicit data such as ratings, collaborative filtering can also use implicit information by observing the habits of users, such as music played, applications downloaded,

websites visited, or books read. To develop a model, two approaches can be used: probability approach and rating prediction. The modeling process is conducted by machine learning techniques such as classification, clustering, and rule-based approach. Based on its characteristics, the model-based also has its advantages and disadvantages. Su and Khoshgoftaar stated that the model-based approach has better predictions than memory based. It is also capable of handling the problem of sparsity and scalability better than memory based. However, the model-based approach requires a great resource, such as time and memory, to develop the model and may lose information when using dimensionality reduction.

📝 2.1.8

Which of the following is a benefit of model-based collaborative filtering over memory-based collaborative filtering?

- It handles data sparsity and scalability better.
- It requires less memory and time to develop.
- It provides more detailed user information.
- It doesn’t require machine learning techniques.

🕮 2.1.9

Strengths of collaborative filtering:

- Serendipitous Recommendations: Collaborative filtering can provide serendipitous recommendations by identifying items that users with similar preferences have liked or rated highly. It can introduce users to new and unexpected items they might not have discovered otherwise.
- Scalability: Collaborative filtering can handle large-scale datasets efficiently. The computation involved in identifying similar users or items can be distributed or parallelized, making it suitable for systems with many users and items.
- Cold-Start Problem: Collaborative filtering can handle the cold-start problem, where there is limited or no information about new users or items. By leveraging the preferences of similar users or items, collaborative filtering can make meaningful recommendations even in the absence of user or item data.

Weaknesses of collaborative filtering:

- Data Sparsity: Collaborative filtering relies on user-item interaction data, and in many cases, such data can be sparse. When users have limited interactions or when the number of items is large, it can be challenging to find sufficient overlap between users' preferences, leading to inaccurate or less diverse recommendations.
- Scalability with Growing User Base: As the number of users grows, the computation required to identify similar users or items increases, impacting the scalability of collaborative filtering. The computational complexity can become a challenge as the user base expands.
- Cold-Start Problem for New Users or Items: While collaborative filtering can handle the cold-start problem to some extent, it still faces challenges when dealing with entirely new users or items for which no or minimal data is available. It can be difficult to provide accurate recommendations without sufficient historical data.

2.1.10

Model-based collaborative filtering builds a model using _____ learning techniques. Unlike memory-based filtering, it can handle _____ in data and process large data sets. However, developing the model requires substantial _____ resources.

- machine
- computational
- sparsity

2.1.11

Examples of collaborative filtering in recommender systems

- **User-based collaborative filtering:** In this approach, recommendations are made based on the similarities between users. The system identifies users with similar preferences and recommends items that similar users have liked or rated highly. For example, if User A and User B have similar tastes and preferences, and User A has rated Item X highly, the system would recommend Item X to User B.
- **Item-based collaborative filtering:** This approach focuses on the similarities between items rather than users. It recommends items based on the associations between items and the user's past interactions. For example, if User A has shown interest in and interacted with Item X, and Item X is like Item Y based on their attributes or previous user interactions, the system would recommend Item Y to User A.
- **Matrix factorization:** Matrix factorization techniques, such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), are commonly used in collaborative filtering. These techniques decompose the user-item interaction matrix into latent factors that capture underlying preferences and relationships between users and items. Recommendations are then generated by predicting missing values in the matrix or estimating user preferences based on the learned latent factors.
- **Neighborhood-Based Collaborative Filtering:** This approach identifies neighborhoods of similar users or items and uses their preferences to make recommendations. User-based neighborhood collaborative filtering looks for users with similar tastes and recommends items that their neighbors have liked. Item-based neighborhood collaborative filtering identifies similar items based on user interactions and recommends items highly associated with the user's previous preferences.

2.1.12

What are the basic types of Collaborative Filtering?

- Memory-based collaborative filtering
- Model-based collaborative filtering
- Procesor-based collaborative filtering
- Memory-based individual filtering

2.2 Content-based filtering

2.2.1

A content-based recommender system seeks to infer a user's characteristics or behavior based on the qualities of the item to which the user responds favorably [Roy 2020].

The next table presents an example of a movie recommendation:

Movies	User 1	User 2	User 3	User 4	Action	Comedy
Item 1	1		4	5	Yes	No
Item 2	5	4	1	2	No	Yes
Item 3	4	4		3	Yes	Yes
Item 4	2	2	4	4	No	Yes

The final two columns, Action, and Comedy, list the film genres. Given these genres, we can now identify which users prefer which genre, allowing us to develop features tailored to that user based on how they respond to films in that genre.

Once we know the user's preferences, we can use the feature vector produced to embed the user in an embedding space and provide recommendations based on their preferences. The user's favorite feature vectors from prior records are combined with the item's feature vectors to generate the similarity metrics (we'll discuss it briefly). The best few are then suggested.

When making suggestions to one person, content-based filtering does not need the data of other users.

📝 2.2.2

What primary data does content-based filtering rely on for recommendations?

- Qualities of items the user has responded to positively
- Data from similar users
- Average ratings across all users
- User demographics and profile information

📝 2.2.3

Which statements are true about content-based filtering?

- It does not require data from other users.
- It uses item feature vectors to calculate similarity metrics.
- It always compares user preferences with those of similar users.
- It relies on demographics to build user profiles.

🕮 2.2.4

Strengths of Content-based Filtering:

- At present time, no user data is required for the creation of suggestions. Content-based filtering does not require user data to generate recommendations, in contrast to collaborative filtering. A content-based filtering system may start recommending pertinent information after a user has searched for, browsed through, and/or purchased a few items. Therefore, it is perfect for businesses without a sizable user base to test it on. It also works effectively for retailers in certain product categories or specialized markets where there is a significant user base but little user interaction.
- Recommendations will be very useful to the user. Content-based recommenders can be highly tailored to the user's tastes, including recommendations for specific products, because the process depends on matching the attributes of a database object with the user's profile. For example, content-based filtering may be able to recognize a specific user's preferences and interests. Content-based filtering is extremely beneficial for companies that have enormous product libraries of a single type, like those that manufacture phones, where suggestions must be based on a range of different features.
- You solve the "cold start" problem. Collaborative filtering might result in a potential cold start issue when a new website or community has few brand-new users and few user connections. Although content-based filtering requires some initial input from users to begin providing recommendations,

the quality of early recommendations is typically better than those made by a collaborative system, which must accumulate and correlate millions of data points before becoming optimized.

Challenges of Content-based Filtering

- There is a lack of both variation and novelty. Recommendations consider more factors than merely relevancy. Think about how much you liked the movie Tenet. Without a sure, you'll enjoy Inside Man as well. It's possible that you already know this; in that case, a recommender system won't be necessary. As a result, recommendation engines need to produce a variety of surprising results to be effective.
- Scalability is difficult. Each new product, service, or piece of knowledge must have its features documented and categorized. Scalability can be challenging and time-consuming since attribute assignments are laborious and never-ending.
- The traits could be unreliable or inconsistent. The quality of suggestions based on content is determined by the level of competence of the experts who categorize items. Many of the millions of things that require qualities may be misclassified since they may be arbitrary. There must be a system in place to guarantee that attributes are applied accurately and consistently. A content-based recommender system won't work properly otherwise.

2.2.5

What is one strength of content-based filtering?

- It doesn’t require user data to begin making recommendations.
- It requires large amounts of user data to generate suggestions.
- It provides a limited variety of suggestions.
- It can’t be applied to businesses with specialized markets.

2.2.6

What are the challenges of Content-based filtering?

- There is a lack of both variation and novelty.
- Scalability is difficult.
- The traits could be unreliable or inconsistent.
- It automatically provides diverse recommendations.

📖 2.2.7

Examples of content-based filtering in recommender systems

- **eCommerce Recommendations**: In eCommerce platforms like Amazon or eBay, content-based filtering is used to recommend products to users based on their previous browsing and purchasing history. For example, if a user frequently purchases electronic gadgets, the system will recommend similar items, such as new smartphones, cameras, or accessories that share similar features (e.g., brand, type, or specifications). This allows the platform to provide personalized product suggestions tailored to the user's preferences and interests without needing extensive user interaction data.
- **Media Recommendations**: Content-based filtering is widely used in media services like Netflix, Spotify, or YouTube. These platforms recommend movies, music, or videos based on the content users have interacted with before. For example, if a user frequently watches action movies or listens to rock music, the system will suggest other titles within those genres. The recommendation is made by analyzing the attributes of the content, such as genre, director, cast, or musical style, and matching them with the user's past preferences.
- **Video Games and Stores**: In video game recommendation platforms like Steam or Epic Games Store, content-based filtering helps suggest games based on the genres, gameplay styles, or themes the user enjoys. If a user often plays action-adventure games, the system will recommend similar games, such as those within the same genre or with similar mechanics. These suggestions rely on identifying key features of games (e.g., action, puzzle-solving, multiplayer) that the user has shown interest in.
- **Location-Based Recommendations**: Location-based services, such as Yelp or TripAdvisor, use content-based filtering to recommend restaurants, hotels, or attractions based on a user's location and preferences. For example, if a user frequently checks for Italian restaurants in their area, the system will recommend similar restaurants nearby, considering factors such as cuisine, ratings, and price range. The recommendations are personalized based on the content of the places the user has shown interest in, and this system can be especially useful for users exploring new places or seeking specific types of services in their vicinity.

📝 2.2.8

Which of the following is an example of content-based filtering in eCommerce?

- Recommending products based on the user’s past purchases.
- Recommending products based on similar users’ preferences.
- Offering discounts to customers who purchase frequently.
- Suggesting random products without analyzing the user’s history.

📝 2.2.9

Which of the following platforms use content-based filtering for recommendations?

- Netflix (movies based on genre or past preferences).
- Spotify (music recommendations based on listening history).
- Yelp (restaurant recommendations based on past choices).
- Facebook (friend suggestions based on likes).

2.3 Hybrid approaches

🕮 2.3.1

Hybrid filtering combines different filtering techniques to overcome the limitations of individual methods and enhance the recommendation process. The cold start, overspecialization, and sparsity problems are common issues with collaborative and content-based filtering, and hybrid filtering aims to address these by leveraging the strengths of both approaches. By combining collaborative filtering and content-based filtering, hybrid methods can improve the precision and diversity of recommendations.

In a hybrid filtering system, collaborative filtering relies on user interactions and preferences to recommend items based on the behavior of similar users. Meanwhile, content-based filtering recommends items based on the features or characteristics of the items themselves, like genre, brand, or specifications. The hybrid method seeks to merge these strategies in different ways to maximize recommendation quality. There are several strategies to implement hybrid filtering:

1. **Individually Apply Both Approaches**: In this strategy, collaborative and content-based filtering methods are applied separately, and their predictions are combined to provide the final recommendations. The idea is to take advantage of the unique strengths of both methods, combining the collaborative insights with content-specific knowledge.
2. **Create a Broad Consolidative Model**: This approach integrates both collaborative filtering and content-based characteristics into a single unified model, using both user preferences and item features simultaneously. This method aims to combine the benefits of both types of filtering into one system, enhancing recommendation accuracy.
3. **Integrate Content-based Characteristics into Collaborative Filtering**: In this hybridization, content-based features (like item attributes) are incorporated into the collaborative filtering process. This helps to overcome the cold start problem by using item content when there is limited user interaction data.

4. **Incorporate Collaborative Characteristics into Content-based Approaches**: This method integrates collaborative insights into a content-based filtering approach. For instance, a system could suggest items that are similar to those liked by users with similar preferences, thereby blending content characteristics with collaborative knowledge.

Hybrid filtering aims to overcome the weaknesses inherent in individual filtering techniques, offering more robust, personalized, and accurate recommendations. By combining these strategies, hybrid systems provide a way to deliver better results, even in scenarios where one method alone might fall short.

📝 2.3.2

What is the main goal of hybrid filtering in recommender systems?

- Combining collaborative filtering and content-based filtering to improve recommendation accuracy.
- Using collaborative filtering alone for personalized recommendations.
- Applying content-based filtering only when user data is unavailable.
- Recommending random items based on user history.

📝 2.3.3

Which of the following are ways to implement hybrid filtering in recommender systems?

- Individually apply collaborative and content-based methods, then combine their forecasts.
- Integrate content-based characteristics into collaborative filtering.
- Create a broad consolidative model that combines both filtering approaches.
- Only use collaborative filtering in the presence of user data.

🕮 2.3.4

Examples of hybrid filtering in recommender systems

Hybrid recommender systems combine multiple recommendation techniques to take advantage of their individual strengths, improving the overall recommendation quality. Below are some key examples of hybrid filtering approaches:

1. **Weighted Hybrid:** In this method, the recommendations from different techniques (e.g., collaborative filtering and content-based filtering) are combined using weighted averages or scores. Each technique's output is given a weight based on its performance, and the final recommendation is a weighted combination of all the techniques' predictions.
2. **Switching Hybrid:** This approach dynamically selects one recommendation technique over another based on certain criteria, such as the user's history or data availability. For example, when there is little user interaction data (cold start problem), a content-based filtering approach might be used, whereas collaborative filtering might be applied once sufficient data has been gathered.
3. **Feature Combination:** In this method, features extracted from both collaborative and content-based techniques are combined to enhance recommendation accuracy. The idea is to enrich the feature set used to predict user preferences by integrating data from multiple sources, leading to a more robust recommendation.
4. **Cascade Hybrid:** In a cascade approach, one technique is used to pre-filter the item set, and then another technique is applied to the pre-filtered set. For example, content-based filtering might first narrow down a broad set of items, and then collaborative filtering can further refine the recommendations based on user similarities.
5. **Meta-level Hybrid:** In this approach, recommendations from different techniques are treated as "meta-level" information and used to train a higher-level model. This meta-model learns how to combine the outputs from various recommendation techniques and generates a final recommendation, often leading to improved accuracy.

2.3.5

What are typical applications of hybrid filtering in recommender systems?

- Weighted Hybrid.
- Switching Hybrid.
- Feature Combination.
- Cascade Hybrid.
- Meta-level Hybrid

2.3.6

Which hybrid filtering approach uses a weighted average or scores to combine recommendations from different techniques?

- Weighted Hybrid
- Meta-level Hybrid

- Cascade Hybrid
- Feature Combination

2.3.7

Which of the following are examples of hybrid filtering approaches?

- Weighted Hybrid
- Switching Hybrid
- Memory-based Hybrid
- Feature Combination

2.3.8

In a _____ approach, one recommendation technique is used to pre-filter the item set based on a broad set of criteria, such as content _____ or user _____. After that, another recommendation technique is applied to the _____ set to refine the list of suggestions, ensuring more personalized recommendations are made.

- preferences
- cascade
- pre-filtered
- features

Fuzzy Logic Systems

Chapter 3

3.1 Fuzzy logic principles

🕮 3.1.1

Human decision is mostly based on vague information like "add a pinch of salt", "it's quite hot", "not far", etc. These expressions, though not mathematically precise, are commonly used to convey information in everyday life. However, for computers to process this type of information, it needs to be transformed into a more structured and quantifiable form. This is where fuzzy logic plays a crucial role. Fuzzy logic allows computers to interpret and process vague inputs by assigning a degree of membership to elements within a set, which makes it possible for them to handle data that would typically be too uncertain for traditional binary systems. This process is not achievable through classical Boolean algebra, where data is either true or false, without room for middle ground.

Fuzzy logic has been invented by Prof. Zadeh [Zadeh 1965] and used to describe uncertain systems [Zadeh & Kacprzyk 1992] since the ’60s of the 20th century. Since its inception, fuzzy logic has been used to model and handle uncertain systems that could not be addressed effectively with traditional mathematical approaches. Zadeh’s pioneering work laid the foundation for a new way of dealing with imprecise data, offering a more flexible method of reasoning in various fields, including control systems, artificial intelligence, and decision-making. Zadeh's work also led to the development of fuzzy set theory, which allows for more nuanced and gradual representations of data, improving how machines interpret the world in situations where absolute values may not be available or practical.

Fuzzy logic consists in extending logical operators to fuzzy sets. The fuzzy set theory consists in the introduction of the so-called degree or membership of an element's belonging to a set, which can take on values from the interval <0, 1>, in contrast to classical set theory, where each element either belongs to the set or does not. The degree of belonging of an element (e.g. instantaneous temperature) to a fuzzy set (temperature) can take on any value from zero to one (inclusive). With a value of 0, the element definitely does not belong to the set, 0.2 means hardly, 0.5 maybe, 0.8 almost certainly, and 1 means definitely belonging to the set. This elegantly avoids the situation where, in classical logic, we label a temperature of 25°C as pleasant and 24.9°C as unpleasant.

3.1.2

What is the primary advantage of fuzzy logic over traditional Boolean algebra?

- It can handle vague and imprecise data.
- It allows for precise categorization of data.
- It provides exact, unambiguous values for all data.
- It operates faster than classical logic.

3.1.3

Fuzzy logic has been invented by:

- Lofti Zadeh
- Radim Farana
- Cyril Klimeš
- Constantin, the philosopher

3.1.4

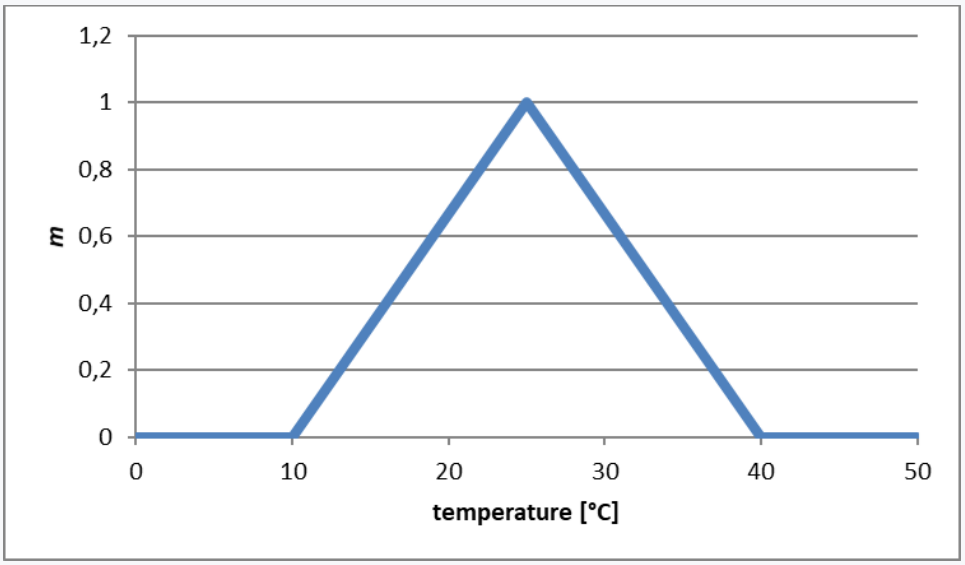
Example

Let's have a set of air temperatures labeled **Pleasant Temperature**. Intuitively, we feel that a temperature of 0°C is definitely not pleasant, 10°C is hardly a pleasant temperature, 20°C is almost certainly pleasant, and 25°C is definitely a pleasant temperature, on the other hand, 35°C will also hardly be pleasant. So we can establish a table of the degree of membership of the actual temperature to the set "**Pleasant temperature**":

temperature	degree of membership *m*	linguistic description
5 °C	0	it's definitely not pleasant
10 °C	0,3	maybe barely
15 °C	0,5	perhaps
20 °C	0,8	almost certainly
25 °C	1,0	most certainly
35 °C	0,5	barely

Obviously, different people may have different opinions about verbal expression and degree of membership.

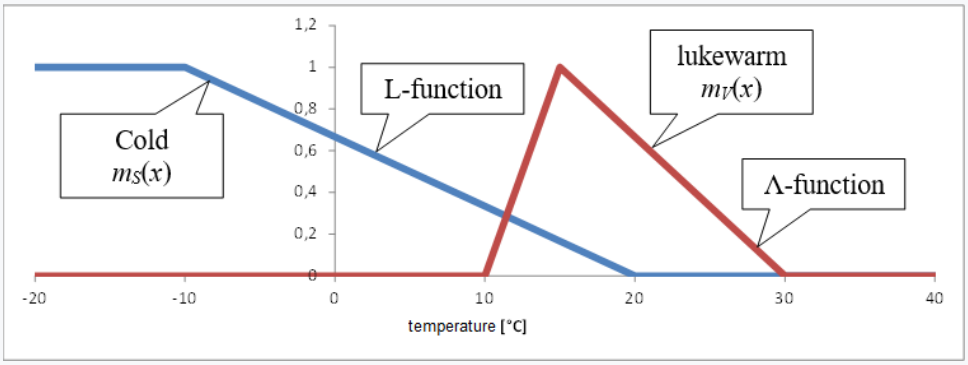
Example of membership function:



At the same time, the degree of membership is not related to the probability of the phenomenon (it also acquires the values 0 ÷ 1), because it does not tell us whether the phenomenon will occur. It only determines with what "strength" a particular value belongs to the selected set. If the function characterizes the degree to which an element belongs to a set, then we refer to these sets as **fuzzy sets**. If we want to use the empirical experience of operators, personnel, and experts, we cannot do without the introduction and use of language variables.

A **linguistic variable** is a variable whose values are expressions of a language. We can interpret the value of a language variable as a fuzzy set. A set of values is referred to as a set of *n* linguistic terms. The meaning of terms is defined by the universe, which we understand as a universal set. E.g. when regulating the temperature of the bath, we can understand the temperature of the water as a language variable named "**Bath temperature**". We measure temperature in centigrades (degrees of Celsius). However, the quantitative expression of the temperature of the bath in the colloquial language does not have to be expressed only in degrees, but in commonly used expressions such as: the bath is ICE COLD, COLD, MOIST, WARM, etc. We can then designate an element from the set of temperatures as the value of the language variable "**Bath temperature**": {icy(L), cold(S), lukewarm(V), hot(H)}.

The linguistic quantification of temperatures introduced in this way using natural language expressions (e.g. cold) represents terms whose meaning is vague and is modeled using fuzzy sets and is defined by the characteristic function $m_S(x)$. The characteristic function $m_S(x)$ is called the membership function $m_S(x)$ for fuzzy sets. It characterizes the degree to which a given element belongs to a given set, from a value of 0, when the element definitely does not belong to the set, to a value of 1, when the element definitely belongs to the set. As an example of membership functions, we present their cold and lukewarm membership functions $m_S(x)$ and $m_V(x)$ for fuzzy sets. The meaning of the terms **cold** and **lukewarm** is modeled by the membership function on a certain temperature interval (universe) in centigrades.



We will explain the membership function of a fuzzy set using the following example:

- If we measure the temperature x = 20°C, then $m_S(x)$ = 0, and certainly this measured temperature does not belong to the term - language value cold.
- If we measure the temperature x = 0°C, then $m_S(x)$ = 0.7, which indicates that this measured temperature belongs to the term - language value cold with the degree of affiliation 0.7.
- If we measure the temperature x = -10°C, then $m_S(x)$ = 1 and it is clear that this measured temperature belongs to the term - language value cold with the degree of affiliation 1.
- If we measure the temperature x = -20°C, then $m_S(x)$ = 1 and this measured temperature also belongs to the term - language value cold with the degree of affiliation 1.
- If we measure the temperature x = +15 °C, then $m_S(x)$ = 0.25, and this measured temperature belongs to the term - linguistic value cold with the degree of affiliation 0.25. But be careful, the membership function $m_V(x)$= 1, from which it follows that this measured temperature also belongs to the set - term lukewarm with membership degree 1.

The process of assigning measured values of input quantities to fuzzy sets using membership functions is referred to as **fuzzification**.

Standard membership functions are used for many purposes: Λ-function (triangular function), L-function (see previous figure), Π-function (trapezoidal function) see Γ-function, S-function, and Z-function.

3.1.5

Which of the following best describes the function of fuzzy sets and membership functions in fuzzy logic?

- Membership functions assign a degree of belonging to an element based on fuzzy set theory.
- Membership functions determine the probability of a phenomenon occurring.
- Fuzzy sets require exact data for proper categorization.
- Membership functions are used to eliminate vagueness in data.

3.1.6

The membership function in fuzzy sets can assign a degree of membership to an element that lies between 0 and 1, indicating the strength of the element's belonging to the set.

- True
- False

3.1.7

If a temperature of 15°C is measured, and the membership function for "cold" returns 0.25, while the membership function for "lukewarm" returns 1, how would the temperature be categorized according to fuzzy logic?

- The temperature would belong to both the "cold" and "lukewarm" sets, with degrees of membership 0.25 and 1, respectively.
- The temperature would belong to the "cold" set with a degree of 1.
- The temperature would belong to the "lukewarm" set with a degree of 0.25.
- The temperature would not belong to either set.

3.1.8

Basic operations with fuzzy sets

Fuzzy sets can be considered a generalization of classical crisp sets. A classical set can be considered a special case of a fuzzy set, whose membership function takes only the values 0 and 1.

On fuzzy sets, we can use operations similar to those we used on classical sets. For our purposes, we will only list the three basic operations (complement, intersection, and union).

Complement of fuzzy set, the complement of set A, C = NOT A:

$$m_C(x) = 1 - m_A(x)$$

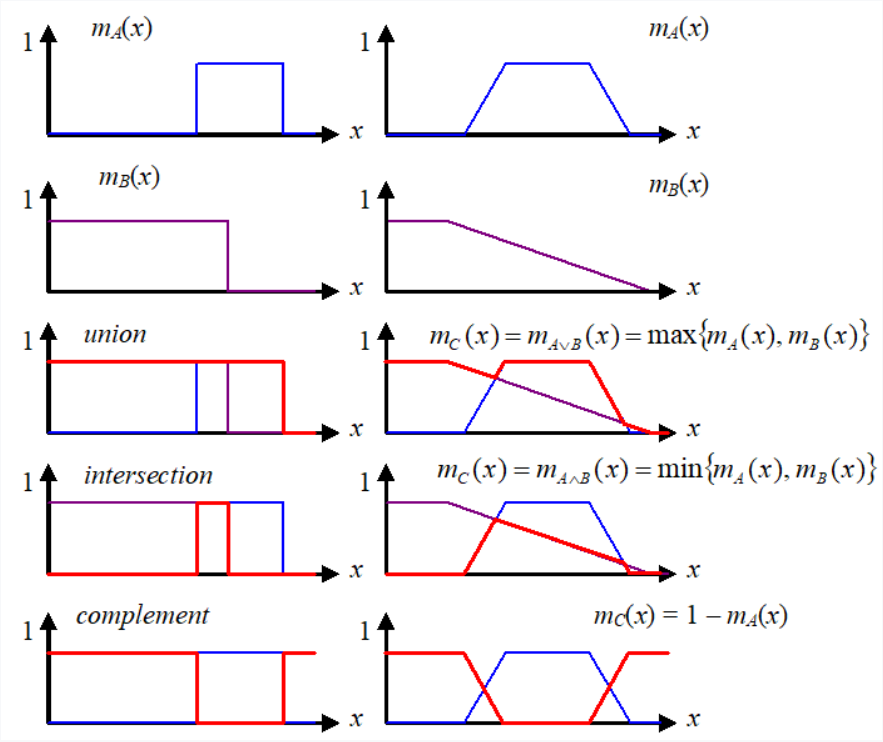
Intersection of fuzzy sets (logical product) C = A AND B

$$m_C(x) = m_{A \wedge B}(x) = \min\{m_A(x), m_B(x)\}$$

Union of fuzzy sets (logical sum) C = A OR B

$$m_C(x) = m_{A \vee B}(x) = \max\{m_A(x), m_B(x)\}$$

These operations are shown in the figure:



3.1.9

We know three basic operations with fuzzy sets, they are:

- Complement, intersection, and union
- Compartment, international, and unia
- Concatenation, interruption, and unique

3.2 Fuzzy rules

3.2.1

Fuzzy logic enables reasoning with imprecise or vague information, using fuzzy rules to make inferences. In general, logical inference involves evaluating decision rules, which are typically presented as conditional statements in the form of.

```
IF ... THEN ...
```

These rules guide the reasoning process by expressing relationships between inputs and outputs, which are not always precise but can still provide valuable information for decision-making. In fuzzy logic, these rules are used to infer conclusions based on fuzzy sets, where the membership of an element to a set is not binary but can vary in degree between 0 and 1.

The condition for inference in fuzzy logic is usually represented by a fuzzy implication statement:

```
IF  THEN
```

Here, both the antecedent (the "IF" part) and the consequent (the "THEN" part) are fuzzy statements. The antecedent describes a condition that is typically a compound statement involving one or more fuzzy expressions. These parts are bound together by logical conjunctions (such as AND, OR), forming a condition that can be partially true or false depending on the degree of membership of the elements. The consequent part of the rule describes the output or conclusion of the fuzzy system, which can be interpreted as a fuzzy set with varying degrees of truth.

In practice, the "IF-THEN" fuzzy rules are applied to map input values to output decisions, allowing systems to reason with imprecise or incomplete data. These rules are essential for decision-making in fuzzy inference systems, such as those used in control systems, image processing, and pattern recognition. The flexibility of fuzzy rules, where both the conditions and the results are described in terms of fuzzy sets, makes them well-suited for handling real-world uncertainties. These rules enable systems to draw conclusions that are not sharply defined, offering more nuanced insights than classical logic.

3.2.2

What is the primary purpose of fuzzy "IF-THEN" rules in fuzzy logic?

- To infer conclusions based on fuzzy sets.
- To provide precise, binary answers.
- To eliminate uncertainties in data.
- To categorize data into strict categories.

3.2.3

Which of the following are true about fuzzy "IF-THEN" rules?

- The consequent part of the rule is the output of the fuzzy system.
- The antecedent may involve logical conjunctions between multiple fuzzy expressions.
- The antecedent is always a single fuzzy expression.
- The consequent part of the rule is always a precise, non-fuzzy value.

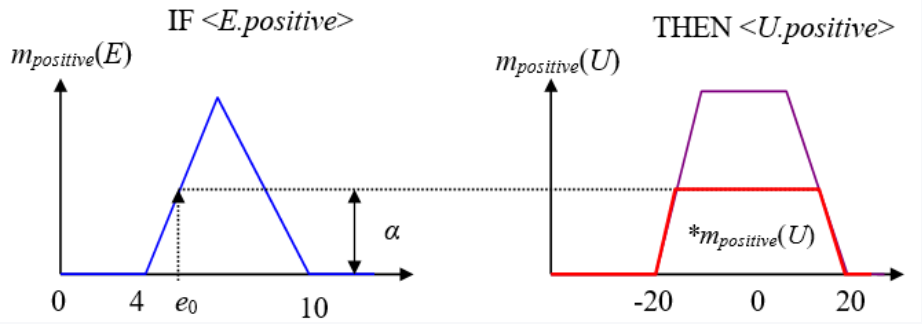
3.2.4

Example

A simple fuzzy rule:

```
IF  THEN
```

In the decision rule, the antecedent contains a linguistic variable E (regulatory deviation), whose value is positive and has the membership function $m_{value}(E)$. The consequent contains a language variable U (action quantity) with a positive value, whose membership function is $m_{positive}(U)$,



If we measure the crisp value of the regulation deviation e_0, then we can use the membership function $m_{positive}(E)$ to subtract the degree of membership α with which the measured value belongs to the set of values *E*.positive. However, our task is to find the corresponding fuzzy set of the consequent for the measured crisp value. The most common procedure for determining this set is based on the logical assumption that the consequence - the consequent can have at most the same degree of membership as the condition - the antecedent. The degree of membership of the measured "crisp" value e_0 therefore determines the level that cuts the output fuzzy set U of the consequent. The membership function of the consequent is then $*m_{kladná}(U)$ (outline in bold).

A generalization of this principle to a two-dimensional antecedent

```
IF (X is positive small) AND (Y is positive medium) THEN (U is
negative medium)
```

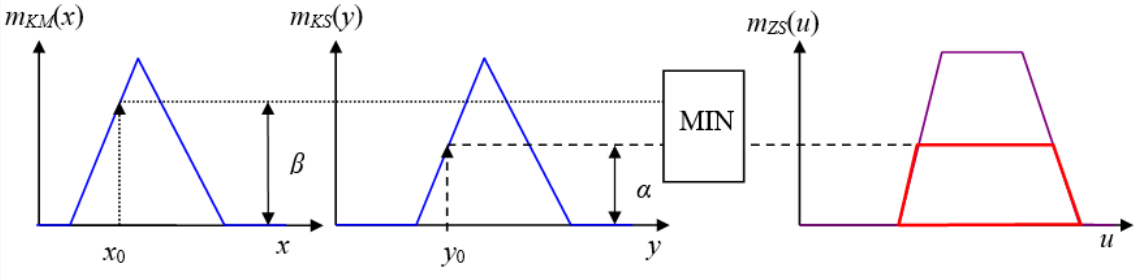
There are several ways to model the meaning of the fuzzy IF-THEN rule, but from the point of view of many applications, the most important is the following method (by Mamdani [Mamdani & Assilian 1975]), which defines the membership function of the consequent as:

$$m_{IM} = \min\{m_A(x_1), m_B(x_2)\}$$

Minimization expresses the fact that the consequence (consequent) can have at most the same degree of membership as the condition (antecedent). Finding the output set for a single rule and a two-dimensional dependency differs depending on whether the AND or OR operator is used. If the operator is AND, then we select the minimum of the corresponding values of the membership functions, see the next figure.

IF AND THEN

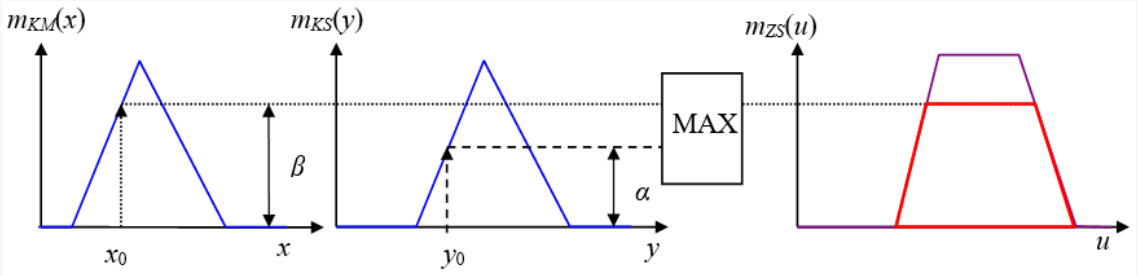
Trimming the membership function of the consequent at level α, which corresponds to the minimum of the two membership functions of the input values.



If the operator is OR, then we select the maximum of the corresponding membership functions, see the next figure.

IF OR THEN

Trimming the membership function of the consequent at the β level that corresponds to the maximum of the two membership functions of the input values.



Finding the output set for two rules and two-dimensional dependence and Mamdani's method is shown in the next figure.

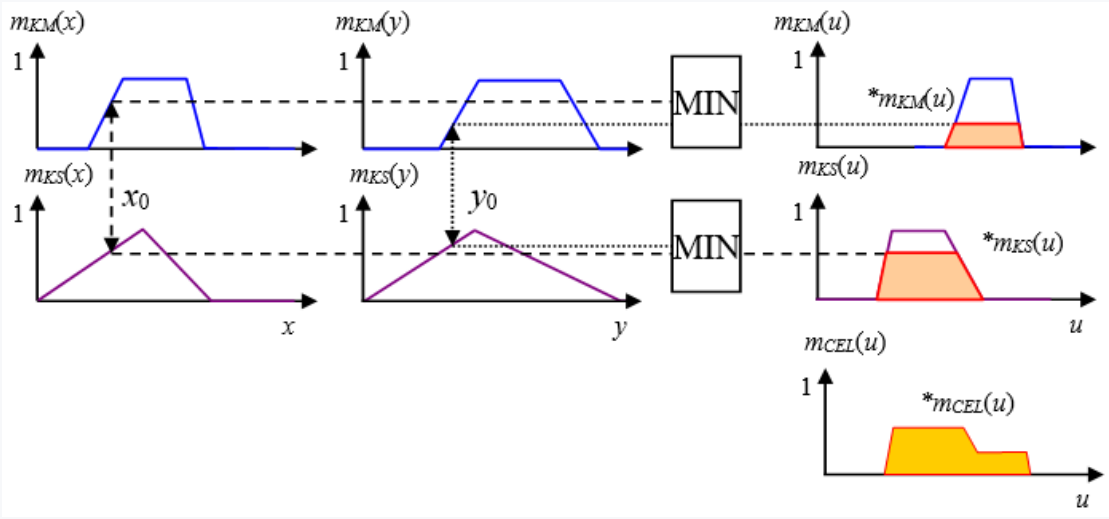
IF AND THEN ELSE IF AND THEN

For two fuzzy IF-THEN rules, their meaning is modeled by these membership functions.

$$\alpha_1 = m_{KM}(x) \wedge m_{KM}(y) = \min\{m_{KM}(x), m_{KM}(y)\}$$
$$\alpha_2 = m_{KS}(x) \wedge m_{KS}(y) = \min\{m_{KS}(x), m_{KS}(y)\}$$

For the consequents of both implications, we get:

$$*m_{KM}(u) = \alpha_1 \wedge m_{KM}(u) = \min\{\alpha_1, m_{KM}(u)\}$$
$$*m_{KS}(u) = \alpha_2 \wedge m_{KS}(u) = \min\{\alpha_2, m_{KS}(u)\}$$



The consequents of both implications $*m_{KS}(u)$ and $*m_{KM}(u)$ determine their partial shares in the size of the action quantity. Intuitively, it is possible to interpret the effect of both partial output terms as their logical sum. Then for the output fuzzy set of both effects, we get:

$$*m_{CEL}(u) = \max\{\min\{\alpha_1, m_{KM}(u)\}, \min\{\alpha_1, m_{KM}(u)\}\}$$

This approach can be extended to any number of decision rules.

3.2.5

In the fuzzy rule

```
"IF <E.positive> THEN <U.positive>"
```

what does the antecedent represent?

- The fuzzy set of regulatory deviation
- The fuzzy set of the action quantity
- The output fuzzy set
- The level of membership for the output fuzzy set

3.2.6

For inferring knowledge, the condition is expressed in the form of an implication of two fuzzy statements mostly called:

- Fuzzy IF-THEN rules
- Fuzzy FOR-THAN rules
- Fuzzy BETWEEN-AND rules
- Fuzzy WHILE-END rules

3.2.7

In a fuzzy rule using the AND operator, what do we do with the membership functions of the antecedents?

- Select the minimum of the corresponding values
- Select the maximum of the corresponding values
- Add the corresponding values together
- Multiply the corresponding values together

3.2.8

In a fuzzy rule using the OR operator, what do we do with the membership functions of the antecedents?

- Select the maximum of the corresponding values
- Add the membership functions
- Select the minimum of the corresponding values
- Subtract the membership functions

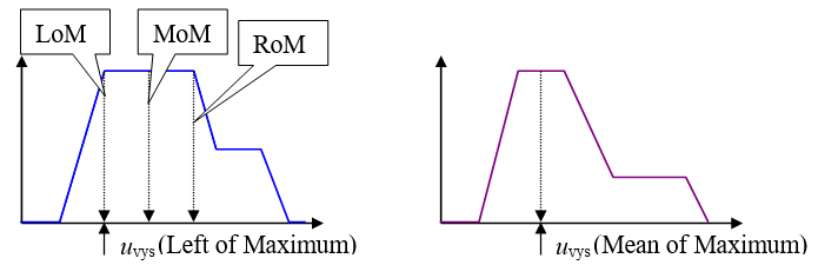
3.2.9

Defuzzification

The result of the action of the block of decision rules is a set of membership functions for individual terms of the output language variables. The membership function of the output set is given by the union of trimmed membership functions (Mamdani's method). After the practical implementation of action interventions, it is necessary to assign the output language variables a crisp value of the action quantity within the permissible range. This process of "approximating fuzzy terms" by a crisp valued action quantity is called **defuzzification**.

There are a variety of **defuzzification methods**, ranging from empirical verification to heuristic approaches. Nest figure present three often used defuzzification methods:

- **LoM (Left of Maximum):** This method takes the value of the output that is to the left of the peak of the membership function. In other words, it selects the point where the membership function is at its highest value but on the left side. It can be useful when it is important to minimize overshooting the target.
- **MoM (Mean of Maximum):** This method calculates the average value of the crisp outputs corresponding to the maximum value of the membership function. It is particularly useful when the output membership function has a large flat region, as it provides a compromise between the possible values.
- **RoM (Right of Maximum):** Similar to LoM, but this method selects the value to the right of the peak of the membership function. It is chosen when it is important to avoid under-shooting the desired target, ensuring the output is on the higher side of the membership function.



When choosing a defuzzification method, we can choose either method that determines the value of the action variable by calculation as the best compromise (**center of gravity** methods) or methods that search for an acceptable solution (methods of the **most significant maximum**).

- **Center of Gravity** choose if the goal is to generate the most balanced, accurate crisp output from the fuzzy set, especially for continuous systems or systems requiring high precision.
- **Most Significant Maximum (LoM, MoM, RoM)** choose when computational efficiency is a priority or when the fuzzy set's shape is not highly irregular, and you need a more practical solution without the need for high precision.

3.2.10

What is the purpose of defuzzification in a fuzzy logic system?

- To convert fuzzy output terms into a crisp value for practical use
- To combine the membership functions into a single fuzzy set
- To calculate the degree of membership of input variables
- To calculate the maximum value of the membership function

3.2.11

Often used defuzzification methods are:

- Left of Maximum
- Mean of Maximum
- Right of Maximum
- Center of Gravity

3.2.12

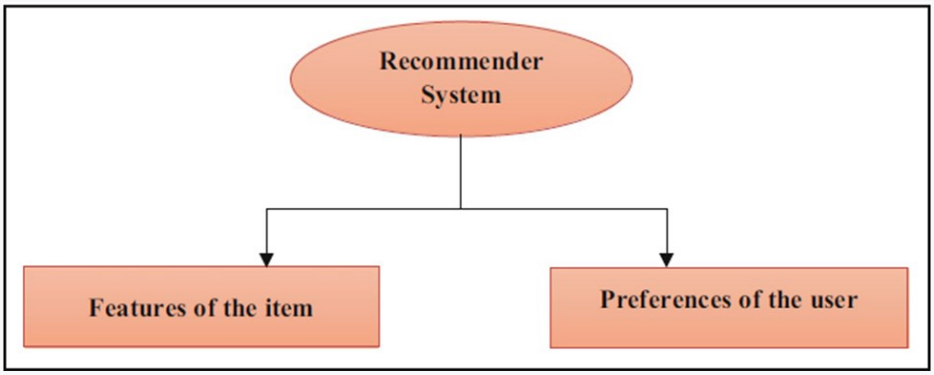
Which defuzzification method is useful when the output membership function has a large flat region?

- MoM (Mean of Maximum)
- LoM (Left of Maximum)
- RoM (Right of Maximum)

3.3 Fuzzy logic in recommender systems

3.3.1

A recommender system considers both the user's preferences and the features of the item [Gupta and Jain, 2018]:



These criteria and their representation affect how well a recommender system performs and operates. These two elements are essential in showing a user how personalized a service is.

However, these variables are arbitrary and ill-defined. The recommender system will have a difficult time managing the uncertainty as a result. It's also noteworthy to note that these uncertainties affect the goods' attributes to varying degrees (low to high).

A car might, for instance, have very good interiors, okay, not so good, horrible, or awful.

To handle this ambiguity, fuzzy logic in recommender systems is highly advised. In addition to assisting in the logical modeling of vagueness, this offers heuristic recommendations and a cogent framework for recommender system design. In literature, a recommender system has been developed utilizing fuzzy logic. Due to the use of fuzzy logic, the following substantial changes have been made to the recommendation:

- the fuzzy rules help to model the vagueness in user input in an efficient manner with the help of linguistic variables (“like”/“dislike”/“awful”/“good”).
- use of fuzzy logic with other techniques like clustering improves the determination of similar users, by finding the fuzzy similarity degree between two users,
- the closeness between two fuzzy ratings (user input) over a particular product has been understood easily with the help of fuzzy rules.

3.3.2

What are the advantages of fuzzy logic in recommendation systems?

- The fuzzy rules help to model the vagueness in user input in an efficient manner with the help of linguistic variables (“like”/“dislike”/“awful”/“good”).
- Use of fuzzy logic with other techniques like clustering improves the determination of similar users, by finding the fuzzy similarity degree between two users.
- The closeness between two fuzzy ratings (user input) over a particular product has been understood easily with the help of fuzzy rules.
- The fuzzy logic makes the recommender system so complicated, that it is practically unusable for standard customers and they mest ask specialists for their help.

3.3.3

What is the main benefit of using fuzzy logic in a recommender system?

- It helps to handle ambiguity and vagueness in user preferences and item attributes.
- It increases the system's computational speed.
- It eliminates the need for user input.
- It works for binary preferences (like/dislike).

📝 3.3.4

Fuzzy logic in recommender systems helps handle user preferences and item attributes with _____. This is done using linguistic variables like like and _____.

- like
- uncertainty
- good
- awful

🕮 3.3.5

Application of fuzzy logic

Fuzzy logic has applications in a variety of industries, including those that govern the environment, residential appliances, and automobile systems.

Common uses include [Sayantini, 2023]:

- Controlling the height of satellites and spacecraft in the aerospace industry.
- It is also used to regulates the flow of traffic and speed in the automobile systems.
- It is utilized in huge corporate business for personal evaluation and decision assistance systems.
- In the chemical sector, it is also used to regulate the pH, drying, and chemical distillation processes.
- Fuzzy logic is employed in many sophisticated applications of artificial intelligence, including natural language processing.
- Expert systems, one type of contemporary control system, use it heavily.
- Fuzzy logic mirrors human decision-making, but much more quickly. As a result, it is compatible with neural networks.

🕮 3.3.6

Controlling in the aerospace industry

Fuzzy logic plays a critical role in controlling the height of satellites and spacecraft, ensuring that these systems maintain their position with high precision. This process is essential for ensuring that satellites remain in their intended orbit, especially when considering the constant changes in gravitational forces, atmospheric drag, and other external factors. Fuzzy logic controllers can process these imprecise variables and help adjust the satellite's altitude smoothly. For example, a satellite in geostationary orbit may experience slight deviations from its expected position due to solar winds or changes in mass distribution within the spacecraft. Using fuzzy

logic, the control system can make real-time adjustments to maintain the desired height.

In the aerospace industry, traditional control systems often struggle with these imprecise and complex environmental variables. Fuzzy logic, on the other hand, allows for more flexibility in the decision-making process. By modeling the height control system with linguistic variables such as "too high," "acceptable," or "too low," fuzzy logic helps operators fine-tune adjustments to the spacecraft's altitude. It can adjust control variables gradually, much like how a human would make small corrections to achieve the desired outcome.

An example of fuzzy logic in action is the use of fuzzy control in the guidance systems of spacecraft. The fuzzy logic controller uses input from sensors, like altimeters and gyroscopes, and outputs commands to the spacecraft's thrusters. This real-time adjustment ensures that the spacecraft stays within a specific height range, even under changing conditions.

3.3.7

Which of the following are important factors considered in the fuzzy control system for maintaining the height of spacecraft?

- Solar winds
- Gravitational forces
- Air pressure
- Temperature fluctuations

3.3.8

Regulating in automobile systems

Fuzzy logic is also widely used in automobile systems, particularly in regulating the flow of traffic and controlling vehicle speed. Traffic control systems can benefit from fuzzy logic by adapting to changing traffic patterns in real-time. For example, during peak hours, the traffic light systems can adjust the timing of red and green lights to optimize the flow of vehicles, preventing congestion. Fuzzy logic enables these systems to make decisions based on input variables like traffic density, time of day, and weather conditions. It can also adjust vehicle speed in adaptive cruise control systems, where the car adjusts its speed to maintain a safe distance from the vehicle ahead.

In addition to real-time traffic flow adjustments, fuzzy logic can also improve safety. For instance, fuzzy logic is used in systems that monitor vehicle speed, detecting when a car is traveling too fast for the road conditions. A fuzzy controller might

interpret the road conditions as "wet," "slippery," or "clear," and adjust the vehicle's speed accordingly, even without precise input data. These systems help to improve driving safety by making small, gradual changes based on the degree of uncertainty in the input data, just as a human driver might adjust their speed without exact measurements.

An example of fuzzy logic in action is the adaptive cruise control (ACC) system found in modern vehicles. This system uses sensors to detect the distance between vehicles and the speed at which they are traveling. The fuzzy logic controller then adjusts the vehicle's speed, taking into account factors like the relative speed of the vehicle ahead and the driver’s preset desired speed. The system interprets these factors as linguistic variables such as “close,” “far,” “fast,” and “slow,” and calculates the most appropriate speed adjustment for safety and comfort.

3.3.9

Which factors might fuzzy logic consider when regulating vehicle speed in adaptive cruise control?

- Distance to the vehicle ahead
- Weather conditions
- Vehicle weight
- Road surface condition

3.3.10

Personal evaluation and decision assistance systems in corporate businesses

In corporate businesses, fuzzy logic is employed for personal evaluation and decision assistance systems, where decisions need to be made based on imprecise data. For example, human resources departments may use fuzzy logic to assess employee performance based on factors such as work efficiency, communication skills, and team collaboration. These factors are often subjective, making it difficult to assign precise numerical values. Fuzzy logic allows HR professionals to assign linguistic variables like "excellent," "good," or "needs improvement" to these attributes and make decisions based on these fuzzy values.

Fuzzy logic is particularly useful in situations where multiple factors need to be considered simultaneously. For instance, when evaluating job candidates, HR departments might use fuzzy logic to weigh various criteria such as experience, skills, and cultural fit. Instead of making a binary decision, fuzzy logic allows for a range of evaluations, helping decision-makers identify the best candidate based on a combination of factors. This flexibility is valuable in making nuanced decisions that are otherwise difficult to model with traditional logic systems.

An example in the corporate sector is the use of fuzzy decision-making systems in project management. Managers may need to evaluate project risk based on variables such as budget, timeline, and scope. Fuzzy logic helps in aggregating these variables, assigning degrees of importance to each, and arriving at a recommendation that best aligns with the company's objectives. These systems make it easier to handle uncertain or incomplete information, allowing businesses to make more informed, flexible decisions.

3.3.11

What are some factors fuzzy logic might assess in personal evaluation systems?

- Work efficiency
- Experience level
- Gender
- Communication skills

3.3.12

Regulating in the chemical sector

In the chemical sector, fuzzy logic is applied to regulate processes such as pH control, drying, and chemical distillation. These processes often require precise adjustments, but many of the variables involved are difficult to measure with exact accuracy. For example, the pH of a solution may need to be adjusted within a narrow range to ensure the quality of the final product. Fuzzy logic systems can adjust the flow of chemicals in real-time based on imprecise inputs, such as the pH value, the temperature, or the concentration of various substances.

Fuzzy logic also helps in processes like drying, where moisture levels in a material need to be monitored and controlled. Fuzzy controllers can interpret inputs such as humidity levels, airflow, and temperature, and adjust the drying process accordingly. This is crucial in industries like food processing, where the degree of dryness affects the quality and shelf life of the product. Similarly, in chemical distillation, fuzzy logic can control the temperature and pressure conditions, adjusting them in real-time based on fuzzy input values to optimize the purity and yield of the desired product.

An example of fuzzy logic in the chemical industry is its use in regulating the distillation process of ethanol. Fuzzy logic can interpret the concentration of ethanol in the distillation column and adjust the temperature and pressure to maintain a consistent product output. This allows for higher efficiency and greater control over the distillation process, ensuring that the final product meets required standards even in the presence of fluctuating input conditions.

3.3.13

Which processes can fuzzy logic help regulate in the chemical sector? (Choose two correct answers)

- pH control
- Drying
- Chemical distillation
- Price calculation

3.3.14

Natural language processing and expert systems

Fuzzy logic is an essential component in many sophisticated artificial intelligence applications, such as natural language processing (NLP) and expert systems. In NLP, fuzzy logic helps process and interpret the vagueness inherent in human language. Words and phrases often have multiple meanings depending on the context, and fuzzy logic allows computers to handle these uncertainties effectively. For example, the word "hot" can mean different things in different contexts - it could refer to temperature, spiciness, or even attractiveness. Fuzzy logic helps NLP systems recognize these nuances and respond appropriately. Expert systems, which are a type of artificial intelligence, also make extensive use of fuzzy logic. These systems are designed to replicate the decision-making ability of human experts in specialized fields. By using fuzzy rules, expert systems can make decisions based on imprecise input. For example, in medical diagnosis, fuzzy logic helps interpret symptoms that may not perfectly match known disease profiles, allowing for more flexible and accurate diagnosis. This is especially useful in fields where uncertainty is prevalent, such as in healthcare or financial forecasting. Fuzzy logic enhances the capabilities of expert systems by allowing them to handle situations where data is incomplete, inconsistent, or vague. It mirrors human decision-making, which is often based on approximations and partial information. The ability to make decisions based on fuzzy data makes expert

systems highly adaptable and valuable in complex decision-making environments. For instance, in environmental monitoring, expert systems using fuzzy logic can interpret varying data about weather, pollution levels, and other factors to make recommendations or trigger alerts. This flexibility makes fuzzy logic essential in applications that require responses to nuanced or ambiguous data, ultimately enhancing the system’s effectiveness.

3.3.15

In NLP, how does fuzzy logic help interpret human language?

- By handling the vagueness in word meanings
- By making language processing faster
- By simplifying text data
- By generating synonyms for words

Fuzzy Expert System

Chapter 4

4.1 Toolbox and project

🕮 4.1.1

MATLAB + Fuzzy logic toolbox

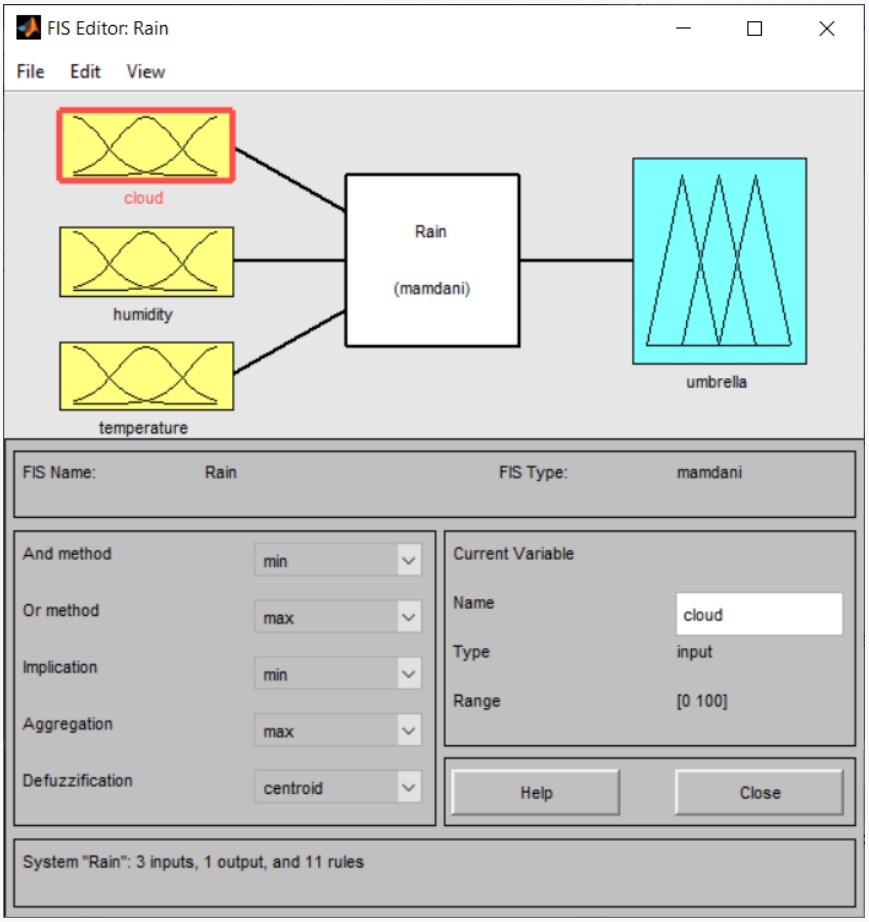
To store the expert knowledge in the form of IF-THEN rules and use this knowledge we construct **Fuzzy Logic Expert Systems.**

To model this expert systems we will use the MATLAB + Fuzzy logic toolbox environment.

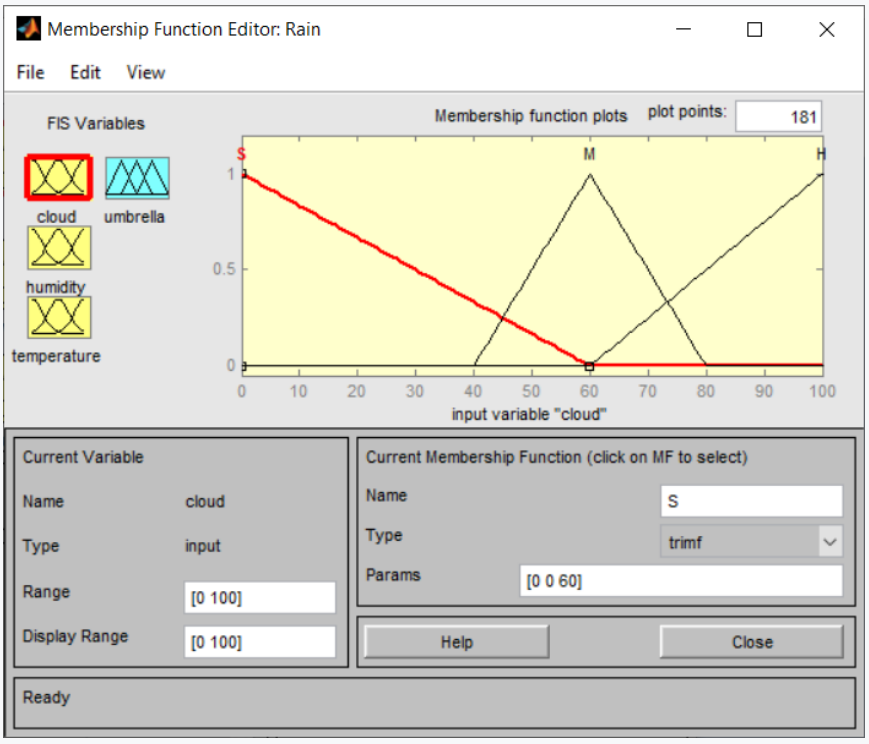
To start the support system in MATLAB we use command:

```
> fuzzy
```

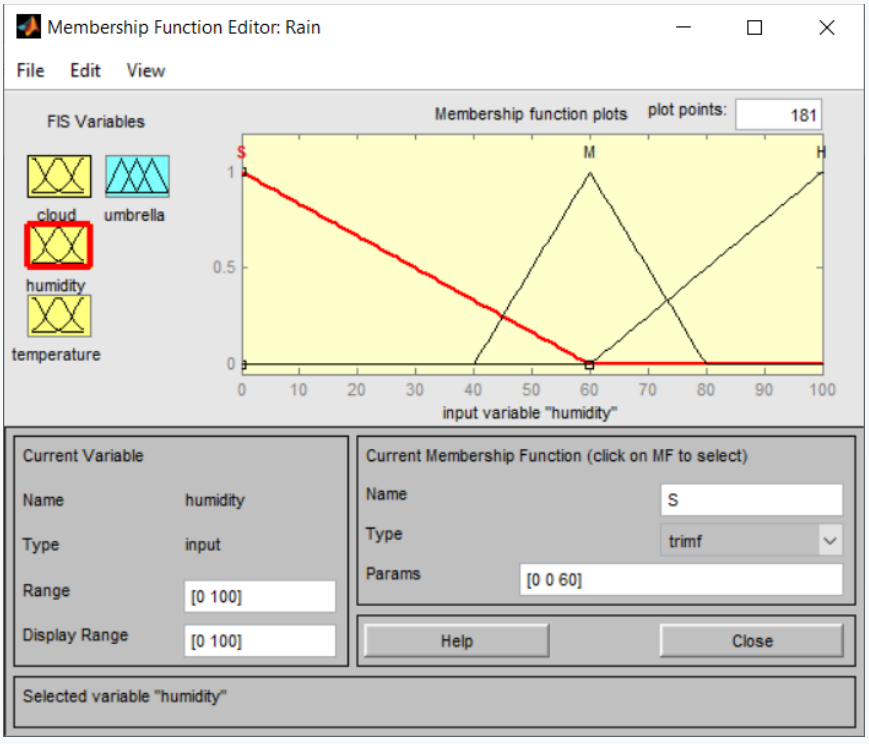
To present some easy example, let us to define expert system, which, based on the knowledge of humidity [%], cloudiness [%] and temperature [°C], will answer the question if you may to take the umbrella.



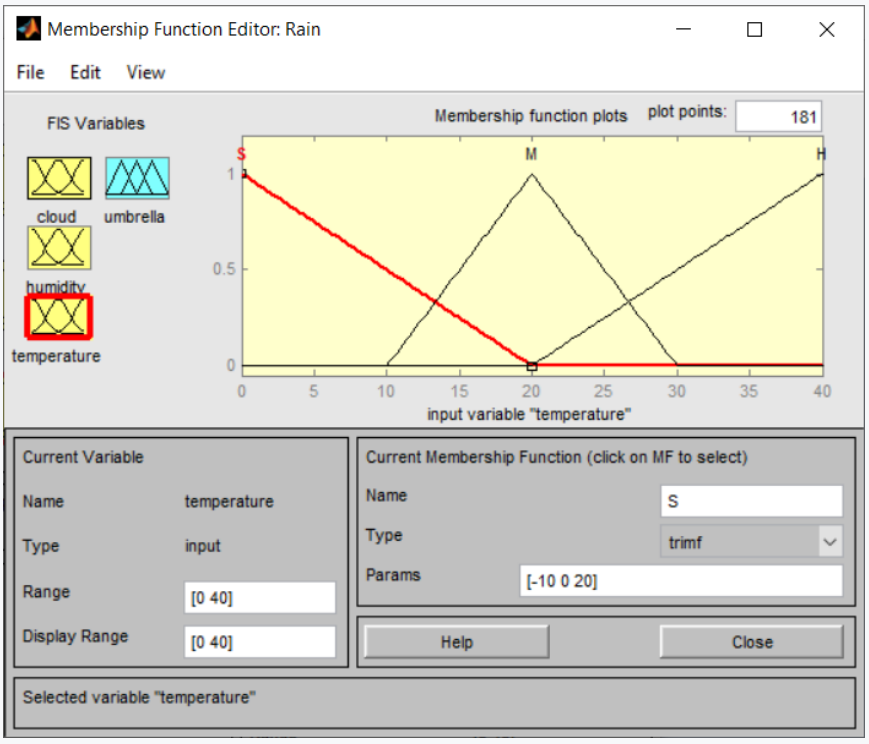
Firstly we define the membership function. For cloudiness we will define 3 fuzzy values (S - small, M - medium and H - high) in the range <0; 100> using the Membership Function Editor:



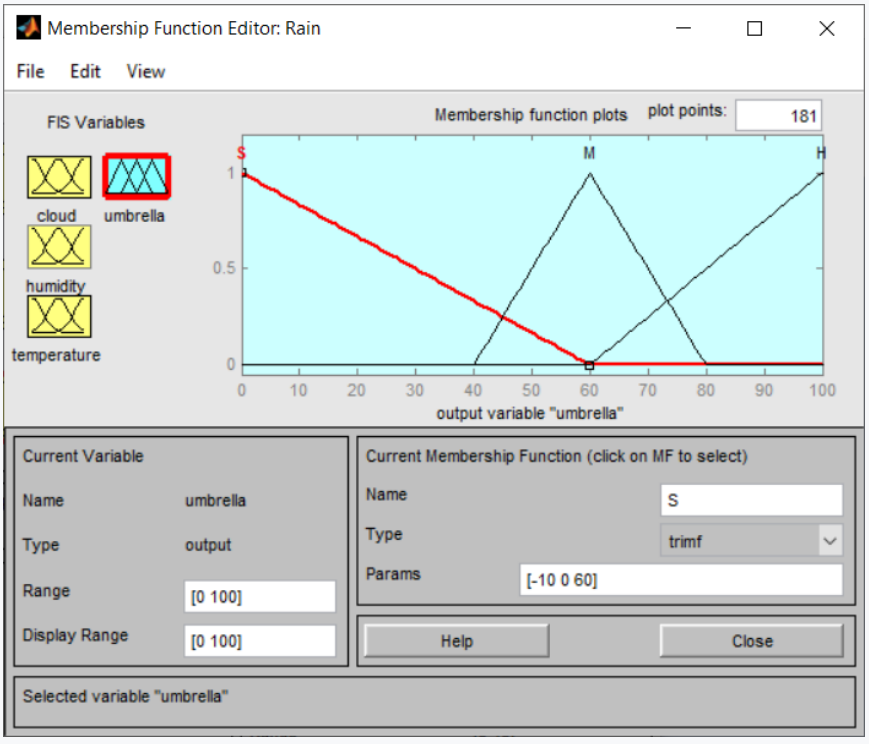
For humidity we will define 3 fuzzy values (S - small, M - medium and H - high) in the range <0; 100> using the Membership Function Editor:



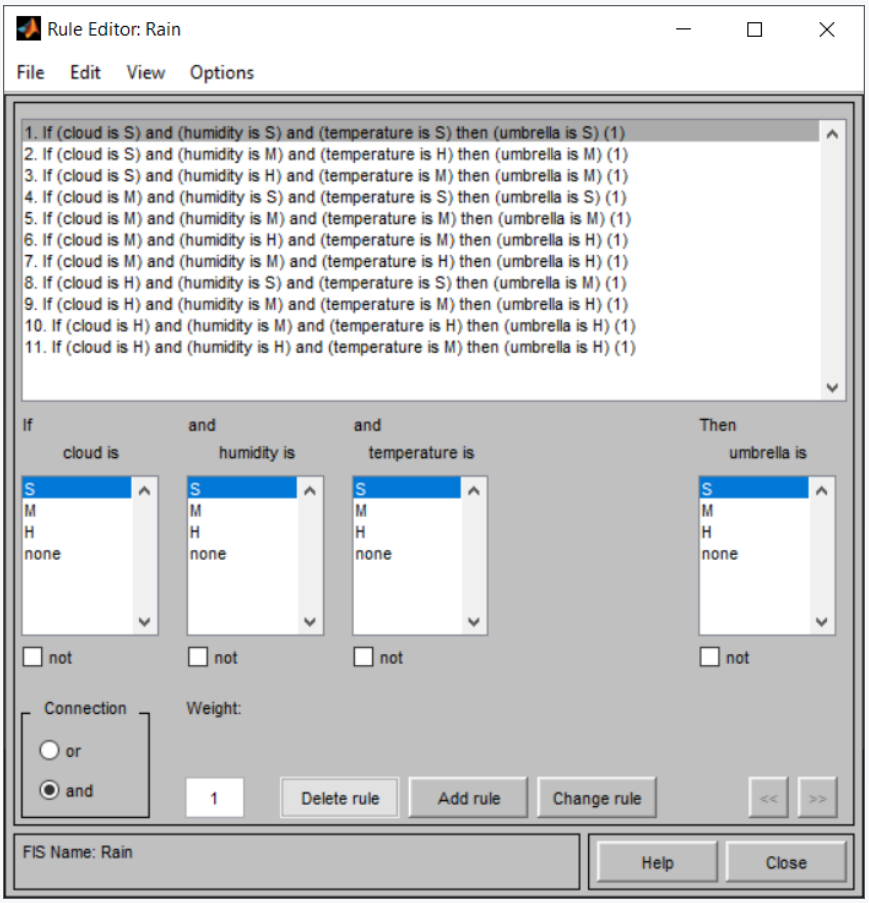
For temperature we will define also 3 fuzzy values (S - small, M - medium and H - high) in the range <0; 40> using the Membership Function Editor:



For output value we will also define 3 fuzzy values as probability to take umbrella (S - small, M - medium and H - high) in the range <0; 100> using the Membership Function Editor:

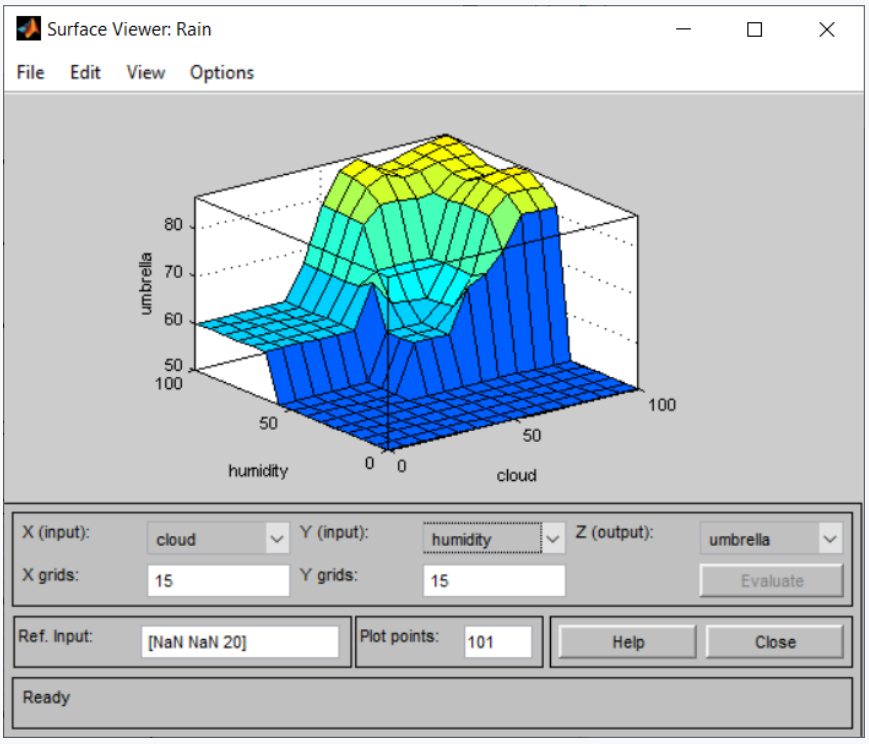


Based on this parameters we can now define the IF-THEN rules:



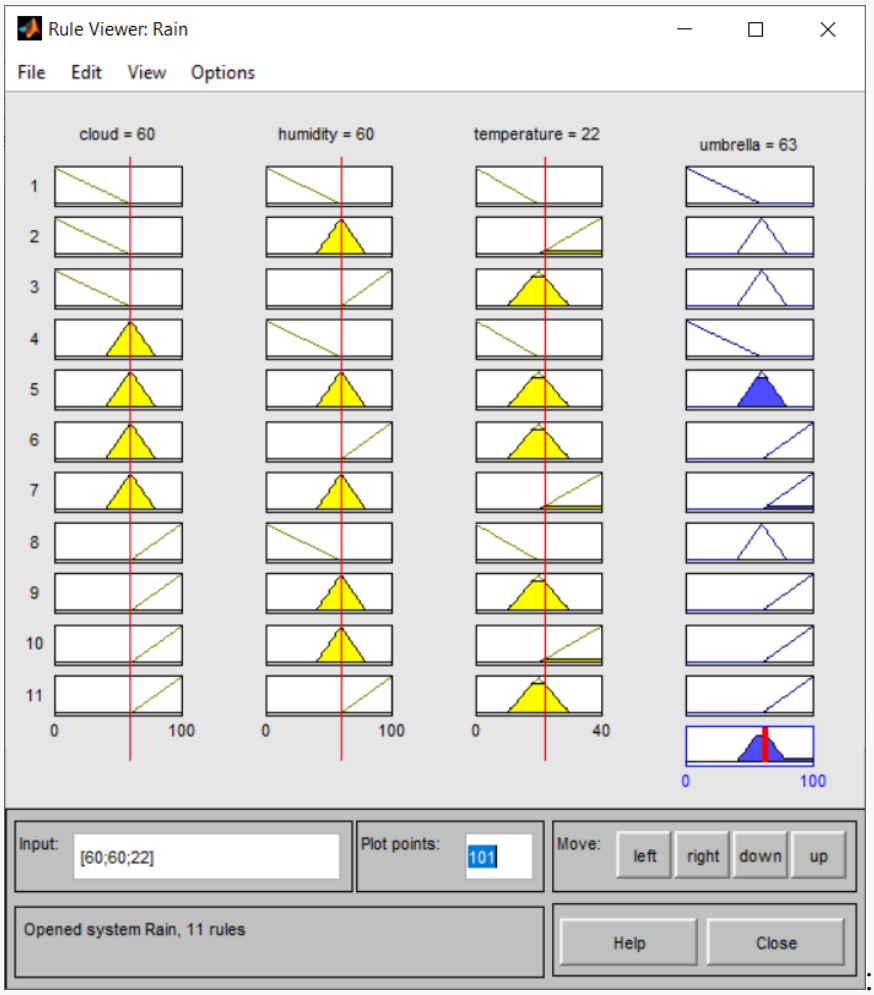
You can see, that there are not defined rules for all combinations of input parameters. This is typical situation when collecting knowledge from concrete experts. Some combinations cannot occur, with some the expert has no experience. We can reduce this problem by combining the knowledge of several experts.

The interaction of the entire set of rules can be well visualized using surface as the interdependence of two input parameters:



This surface can also alert us to inconsistencies in expert opinion or some missing rules.

Once the knowledge base of the expert system is created, we can use it to draw conclusions in specific situations. The Rule Viewer presents it very well. For concrete input (60, 60, 22) it resents not only the result (probability to take the unbrella is 63%), but also presents which rules has been activated and how they contribute to the overall result:



Final fuzzy expert system could be stored as a file, for example Rain.fis for further use. This file is standard text file, so you can edit it:

```
[System]
Name='Rain'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=11
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='cloud'
Range=[0 100]
NumMFs=3
```

```
MF1='S':'trimf',[0 0 60]
MF2='M':'trimf',[40 60 80]
MF3='H':'trimf',[60 100 140]

[Input2]
Name='humidity'
Range=[0 100]
NumMFs=3
MF1='S':'trimf',[0 0 60]
MF2='M':'trimf',[40 60 80]
MF3='H':'trimf',[60 100 140]

[Input3]
Name='temperature'
Range=[0 40]
NumMFs=3
MF1='S':'trimf',[-10 0 20]
MF2='M':'trimf',[10 20 30]
MF3='H':'trimf',[20 40 50]

[Output1]
Name='umbrella'
Range=[0 100]
NumMFs=3
MF1='S':'trimf',[-10 0 60]
MF2='M':'trimf',[40 60 80]
MF3='H':'trimf',[60 100 120]

[Rules]
1 1 1, 1 (1) : 1
1 2 3, 2 (1) : 1
1 3 2, 2 (1) : 1
2 1 1, 1 (1) : 1
2 2 2, 2 (1) : 1
2 3 2, 3 (1) : 1
2 2 3, 3 (1) : 1
3 1 1, 2 (1) : 1
3 2 2, 3 (1) : 1
3 2 3, 3 (1) : 1
3 3 2, 3 (1) : 1
```

4.1.2

Define your problem based on three parameters and construct the fuzzy-expert recommender system to solve it. Was it easy?

4.1.3

Introduction

Innovation and innovative entrepreneurship are nowadays very frequently used terms in various areas of corporate life. The reason for this is that without the ability of companies to adapt to the realities of contemporary life, especially through innovation, these companies have no chance of a long-term and successful existence in the market. For this reason, innovation in companies is given a great deal of attention. Both successful and unsuccessful firms are examined. Moreover, the interest in the study of innovation and innovative entrepreneurship is enhanced by the rapid development of information technology. These technologies are able to generate large amounts of data and, thanks to their high power and technologies such as Data Mining and Artificial Intelligence, to find new connections in the data and new insights into the behaviour of firms and people in firms. The research presented in this paper is moving in the same direction.

Based on the research conducted and the available relevant findings of previous studies, the **main research aim** of the research was to select variables that have a significant or dominant influence on the behavioural patterns of firms leading to desirable innovative entrepreneurship. The **main research area** was chosen as the application of one of the artificial intelligence methods that will be able to identify and evaluate the desired behavioural patterns of firms in an appropriate way. To begin with, a fuzzy approach was chosen to enable a fuzzy understanding of new paradigms of family business development dynamics with a focus on innovative entrepreneurship.

The authors also examined the impact of innovation on firm performance and competitiveness with respect to innovation efficiency. Values of monitored parameters were obtained for each company. Subsequently, significant differences between dependent and independent variables within the research were tested:

Evaluated criterion (dependent variable)		Significant differences – Summary (yes/no) Family business groups compared (independent variable)						Evaluation of the relationship of individual (dependent variables) to the size of the company (yes/total)
		(1, 2)	(1, 3)	(1, 5)	(2, 3)	(2, 5)	(3, 5)	
Innovation focus (OECD)		YES	NO	NO	NO	NO	NO	1/6
Innovation type (OECD)		YES	NO	NO	NO	NO	NO	1/6
Innovation degree (Valenta)		NO	YES	YES	YES	YES	YES	5/6
Branch of businnes		YES	YES	YES	YES	YES	NO	5/6
Economic sector		YES	NO	YES	NO	NO	YES	3/6
Operation – Market size		NO	YES	YES	YES	YES	NO	4/6
Customer type		YES	YES	YES	YES	YES	NO	5/6
Finance resources		YES	YES	YES	YES	YES	NO	5/6
Motivation to innovate	**Main**	NO	NO	NO	NO	NO	NO	0/6
	Revenue/profit	YES	NO	YES	NO	YES	YES	4/6
	Market expansion	NO	YES	YES	YES	YES	NO	3/6
	Adapation	NO	NO	YES	NO	YES	NO	2/6
	Reduction of costs	NO	NO	YES	YES	YES	NO	3/6
	Lack of HR	NO	NO	YES	NO	YES	YES	3/6
	Lack of demand	NO	NO	NO	NO	NO	NO	0/6
	New knoledge	NO	NO	NO	NO	NO	NO	0/6
Results of innovation	**Main**	YES	YES	NO	NO	NO	NO	2/6
	Revenue/profit	NO	YES	YES	YES	YES	NO	4/6
	Market expansion	NO	YES	NO	NO	NO	NO	1/6
	Adapation	NO	NO	YES	NO	NO	NO	1/6
	Reduction of costs	NO	NO	YES	NO	YES	YES	3/6
	Lack of HR	NO	NO	NO	NO	NO	NO	0/6
	Lack of demand	YES	NO	NO	NO	NO	YES	2/6
	New knoledge	NO	NO	YES	NO	YES	YES	3/6
Novelty innovation	**For company**	YES	YES	YES	YES	YES	NO	5/6
	For local market	YES	YES	YES	YES	YES	NO	5/6
	For global market	NO	YES	YES	YES	YES	YES	5/6
Significant differences between category of family businnes (yes/total)		11/27	12/27	18/27	11/27	16/27	8/27	

Thanks to the conducted testing, the real possibility of converting the verbal expression of factors affecting innovation into appropriate decision rules for the missing numerical relationships between various quantities describing the innovative behavior of companies was verified. This represented a basic condition for further work and a real possibility of testing the expert model on selected elements within the sample during the creation of a model of the use of fuzzy logic for evaluating the innovativeness of family firms. A hierarchical fuzzy-expert system was created. At the first level, the classification of the company (to which the company mainly belongs – services (service) or production) is evaluated in terms of its influence on innovativeness. At the second level, the main parameters affecting innovativeness were selected for the model: the size of the family business, the scope of the innovation, the motivation to innovate, the novelty of the innovation for the company, the impact of the innovation on the market.

4.1.4

Which of the following technologies are used to analyze large amounts of data for insights into firm behavior?

- Artificial Intelligence
- Data Mining
- Virtual Reality
- 3D Printing

4.1.5

Innovation and innovative entrepreneurship are crucial for companies because they:

- Ensure long-term market success
- Attract new employees
- Guarantee higher profits every quarter
- Limit the need for technological investment

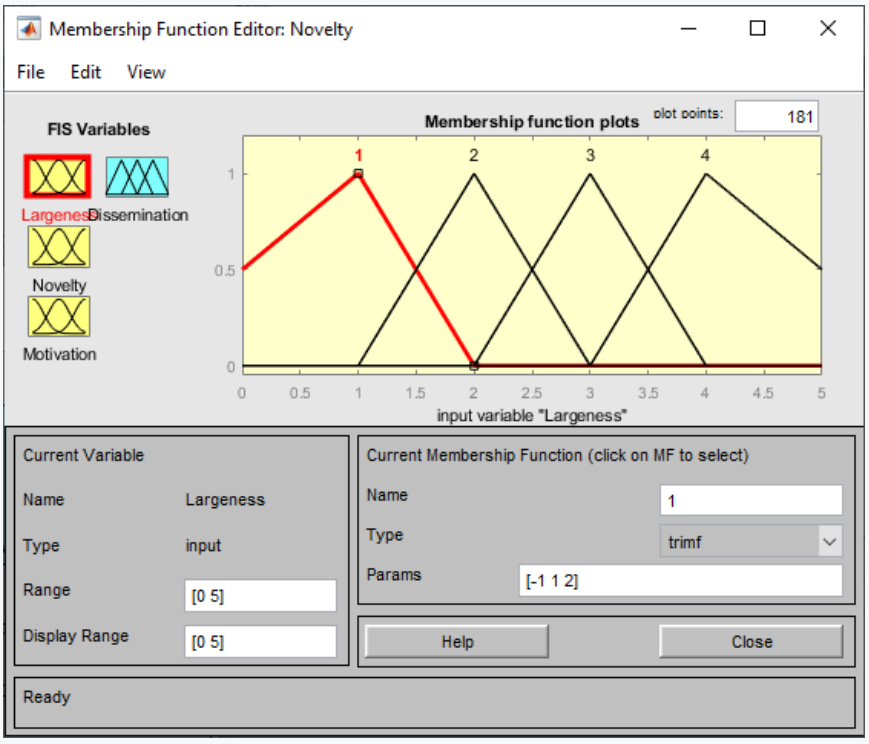
4.1.6

Project: The expert system for family firms from the area of production

The expert system was created on family firms from the area of production, where mutual links were defined, i.e. input and output variables were fuzzified, see next figures. Triangular affiliation functions proved to be suitable. The number of selected fuzzy values seemed suitable for the experts, both in terms of their sufficient number to distinguish individual companies and their status, and the ability to clearly select individual values.

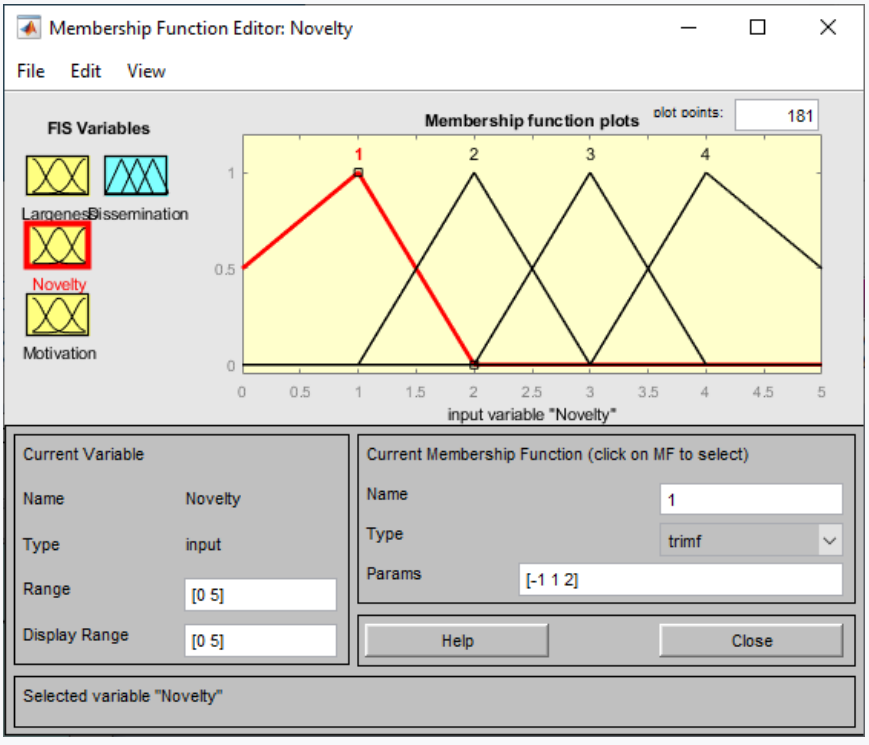
Input parameters fuzzyfication

Family largeness:



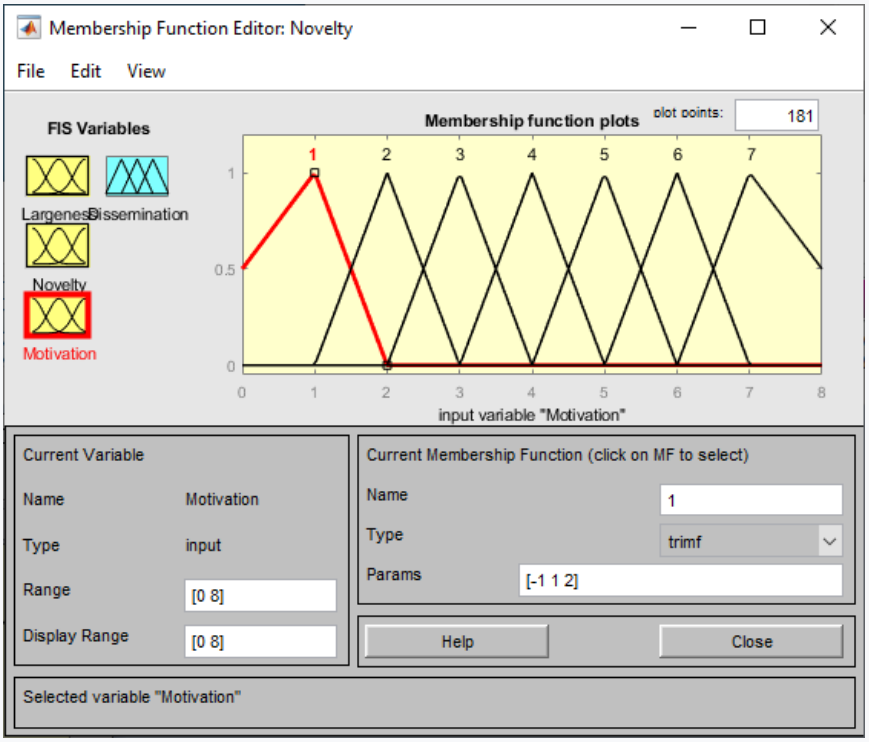
1. 0 - 19 employees (micro enterprises)
2. 20 - 49 employees (small businesses)
3. 50 - 99 employees (smaller medium-sized enterprises)
4. 100 - 249 employees (larger medium-sized enterprises)

Novelty scope:



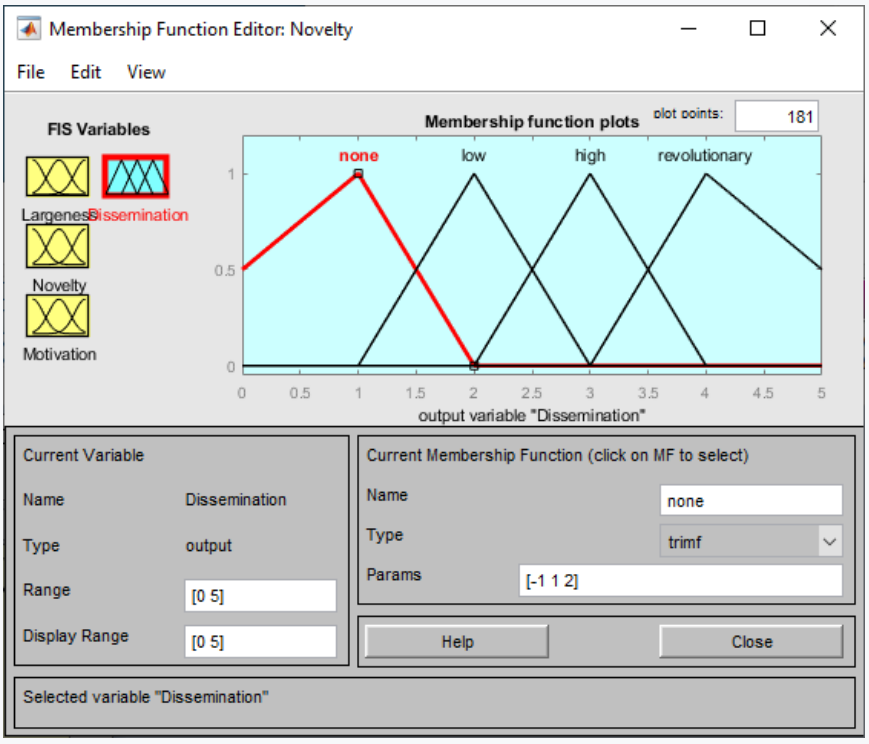
1. RATIONALIZATION INNOVATION: quantum change (expected novelty of order 1)
2. RATIONALIZATION INNOVATION: intensity (expected novelty of order 2)
3. RATIONALIZATION INNOVATION: reorganization (expected novelty of order 3)
4. RATIONALIZATION INNOVATION: quality adaptation (expected novelty of order 4)

Motivation for innovations:



1. Increase in turnover/profit
2. Lack of demand
3. High costs
4. Reduction/increase/improvement etc. of human resources
5. Market expansion
6. Acquisition of new knowledge, need to learn
7. Adapting to innovations in the industry/competitors etc.

As an output value describing the contribution of innovation to the regional market, the dissemination was selected:

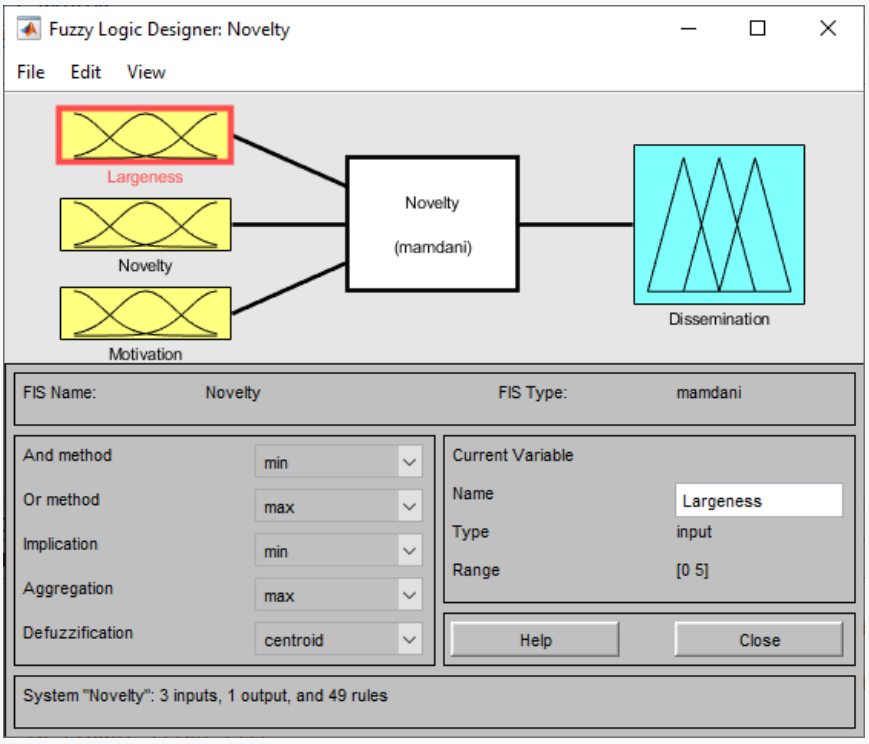


1. Not new to the local market (-)
2. It is partly new for the local market (+)
3. It is completely new to the local market (++)
4. It is absolutely revolutionary for the local market (+++)

🕮 4.1.7

Expert system main structure

The main structure of expert system has been defined in MATLAB environment:



The experts described the dependence between the input quantities and the output quantity and defined individual rules.

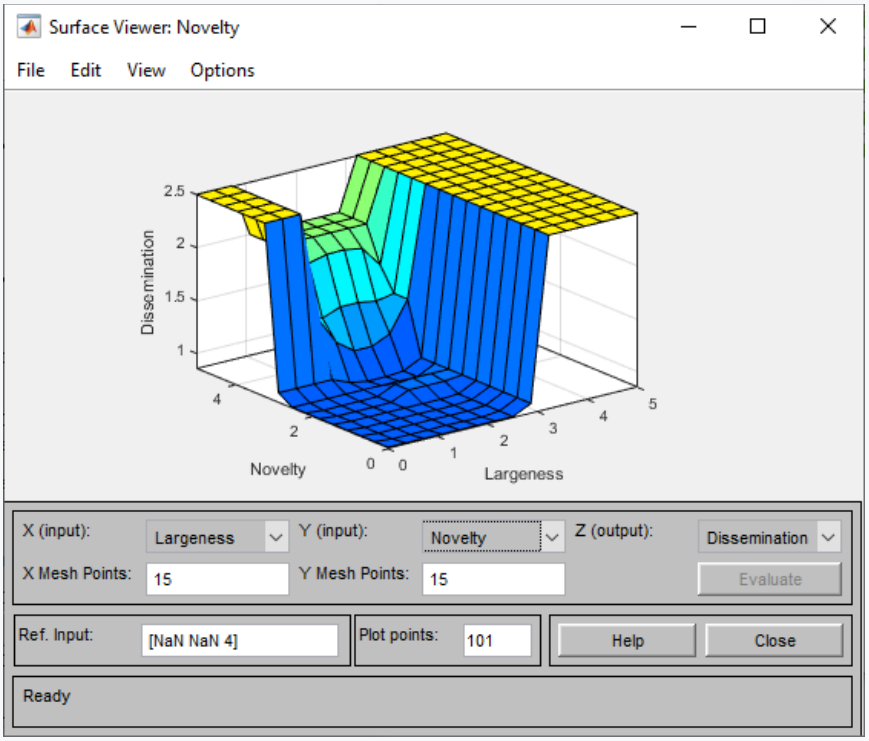
The whole ES definition file is below:

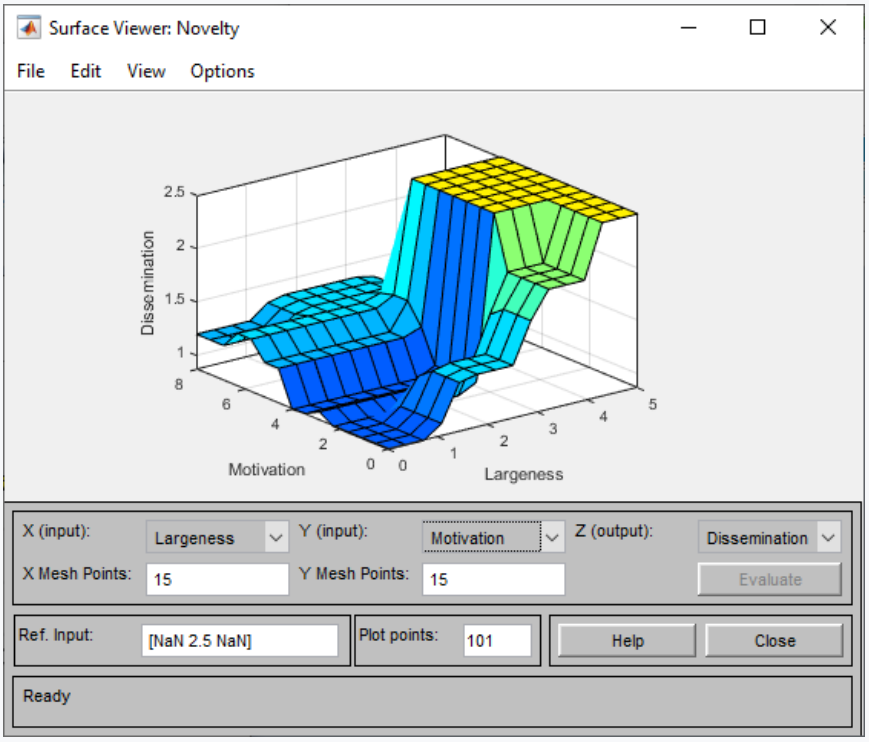
```
[System]
Name='Novelty'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=7
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Largeness'
Range=[0 5]
NumMFs=4
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
```

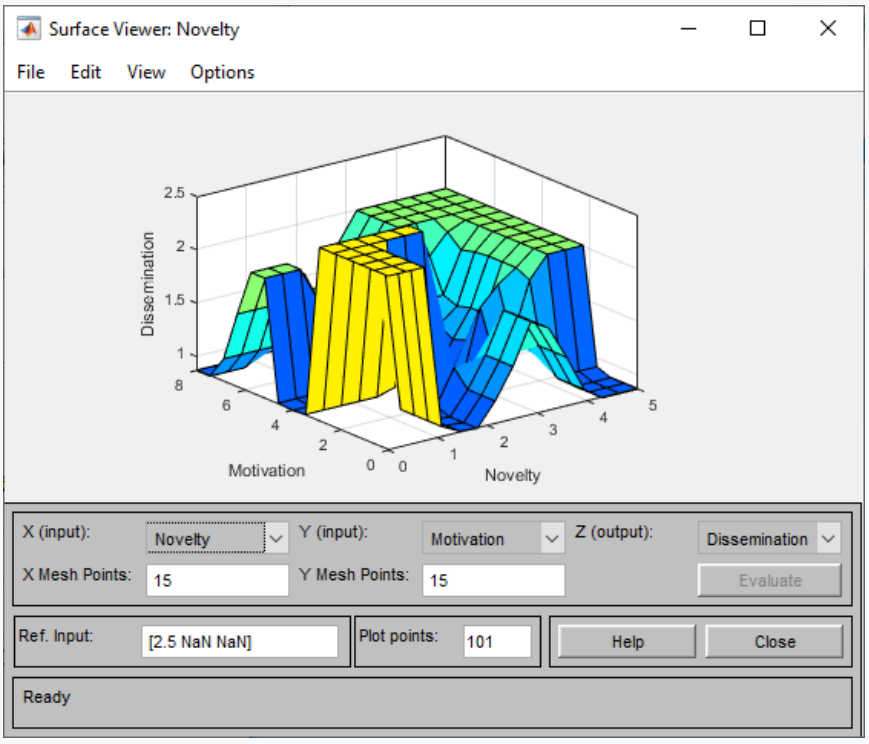
```
MF4='4':'trimf',[3 4 6]

[Input2]
Name='Novelty'
Range=[0 5]
NumMFs=4
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
MF4='4':'trimf',[3 4 6]

[Input3]
Name='Motivation'
Range=[0 8]
NumMFs=7
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
MF4='4':'trimf',[3 4 5]
MF5='5':'trimf',[4 5 6]
MF6='6':'trimf',[5 6 7]
MF7='7':'trimf',[6 7 9]

[Output1]
Name='Dissemination'
Range=[0 5]
NumMFs=4
MF1='none':'trimf',[-1 1 2]
MF2='low':'trimf',[1 2 3]
MF3='high':'trimf',[2 3 4]
MF4='revolutionary':'trimf',[3 4 6]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 1 (1) : 1
1 1 4, 1 (1) : 1
1 1 5, 1 (1) : 1
1 1 7, 1 (1) : 1
1 2 1, 1 (1) : 1
1 2 2, 1 (1) : 1
1 2 4, 1 (1) : 1
1 2 5, 1 (1) : 1
1 2 6, 1 (1) : 1
1 2 7, 1 (1) : 1
```

```
1 3 1, 1 (1) : 1
1 3 2, 1 (1) : 1
1 3 5, 1 (0.5) : 1
1 3 5, 2 (0.5) : 1
1 3 6, 2 (1) : 1
1 3 7, 1 (0.5) : 1
1 3 7, 2 (0.5) : 1
1 4 1, 1 (0.5) : 1
1 4 1, 2 (0.5) : 1
1 4 5, 2 (1) : 1
1 4 6, 1 (0.5) : 1
1 4 6, 2 (0.5) : 1
1 4 7, 1 (1) : 1
2 1 4, 1 (1) : 1
2 1 6, 2 (1) : 1
2 1 7, 1 (1) : 1
2 2 1, 1 (1) : 1
2 2 5, 1 (1) : 1
2 2 6, 1 (1) : 1
2 2 7, 1 (1) : 1
2 3 1, 2 (1) : 1
2 3 2, 1 (1) : 1
2 3 4, 1 (1) : 1
2 3 5, 1 (0.5) : 1
2 3 5, 2 (0.5) : 1
2 3 6, 2 (1) : 1
2 3 7, 2 (1) : 1
2 4 1, 1 (1) : 1
2 4 4, 2 (1) : 1
2 4 6, 2 (1) : 1
2 4 7, 2 (1) : 1
3 2 7, 1 (1) : 1
3 2 7, 2 (1) : 1
3 3 1, 2 (1) : 1
3 3 7, 2 (1) : 1
3 4 3, 2 (1) : 1
3 4 7, 2 (1) : 1
4 2 7, 1 (1) : 1
```

The ES environment is presented in the next figures:







It is evident, that many parts of the parameter space are not covered.

📖 4.1.8

Completing the expert system

Based on the analysis of the ES surface, it was necessary to specifically ask experts about the rules covering all parts of the space.

The resulting expert system description is below:

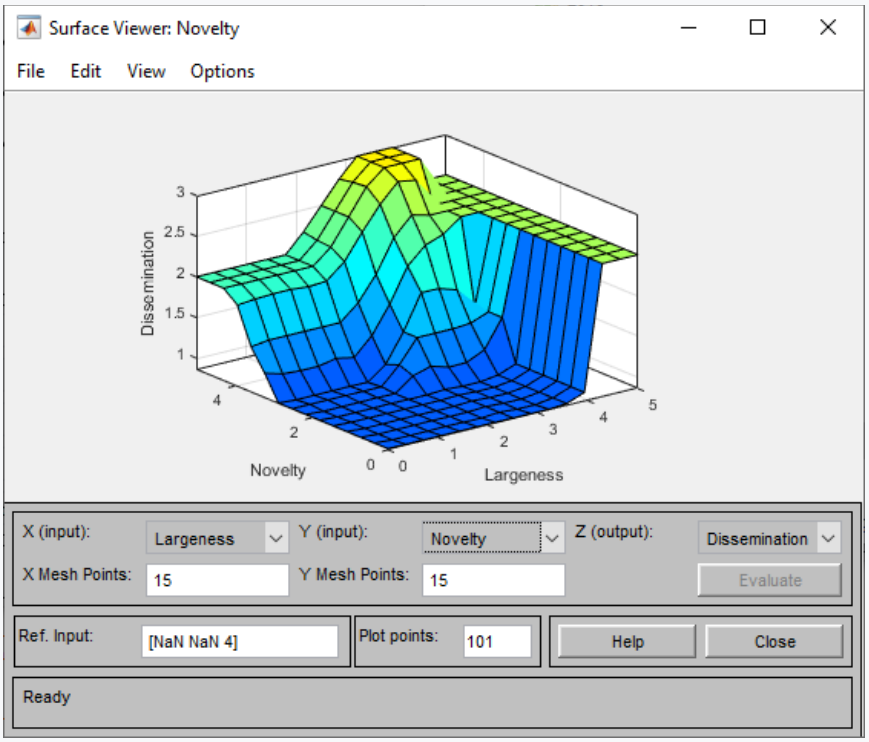
```
[System]
Name='Novelty'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=7
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
```

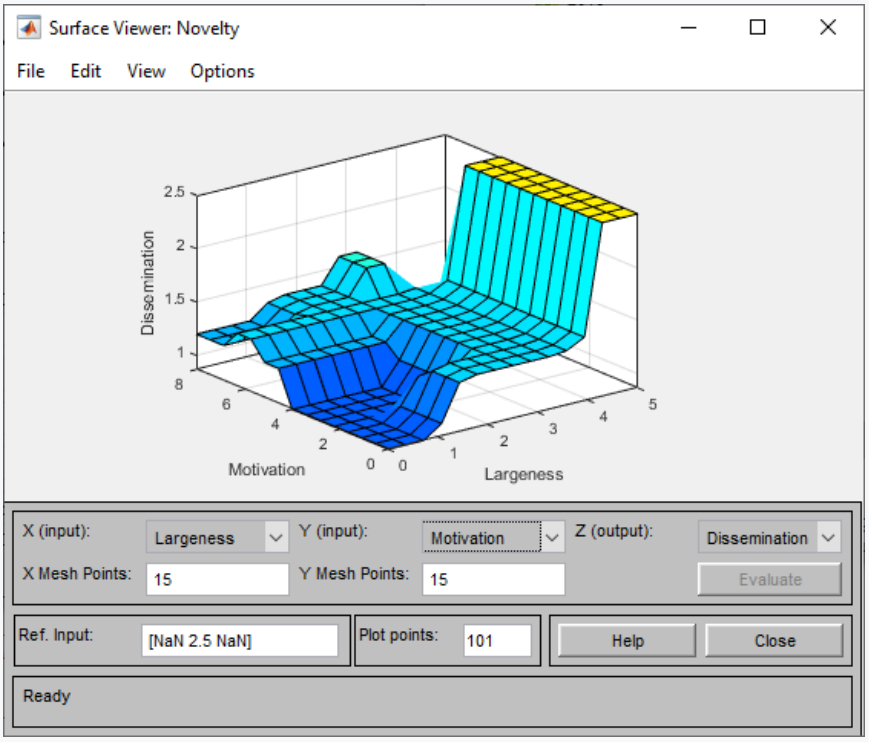
```
Name='Largeness'
Range=[0 5]
NumMFs=4
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
MF4='4':'trimf',[3 4 6]

[Input2]
Name='Novelty'
Range=[0 5]
NumMFs=4
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
MF4='4':'trimf',[3 4 6]

[Input3]
Name='Motivation'
Range=[0 8]
NumMFs=7
MF1='1':'trimf',[-1 1 2]
MF2='2':'trimf',[1 2 3]
MF3='3':'trimf',[2 3 4]
MF4='4':'trimf',[3 4 5]
MF5='5':'trimf',[4 5 6]
MF6='6':'trimf',[5 6 7]
MF7='7':'trimf',[6 7 9]

[Output1]
Name='Dissemination'
Range=[0 5]
NumMFs=4
MF1='none':'trimf',[-1 1 2]
MF2='low':'trimf',[1 2 3]
MF3='high':'trimf',[2 3 4]
MF4='revolutionary':'trimf',[3 4 6]

[Rules]
1 1 1, 1 (1) : 1
1 1 2, 1 (1) : 1
1 1 3, 1 (1) : 1
1 1 4, 1 (1) : 1
1 1 5, 1 (1) : 1
```

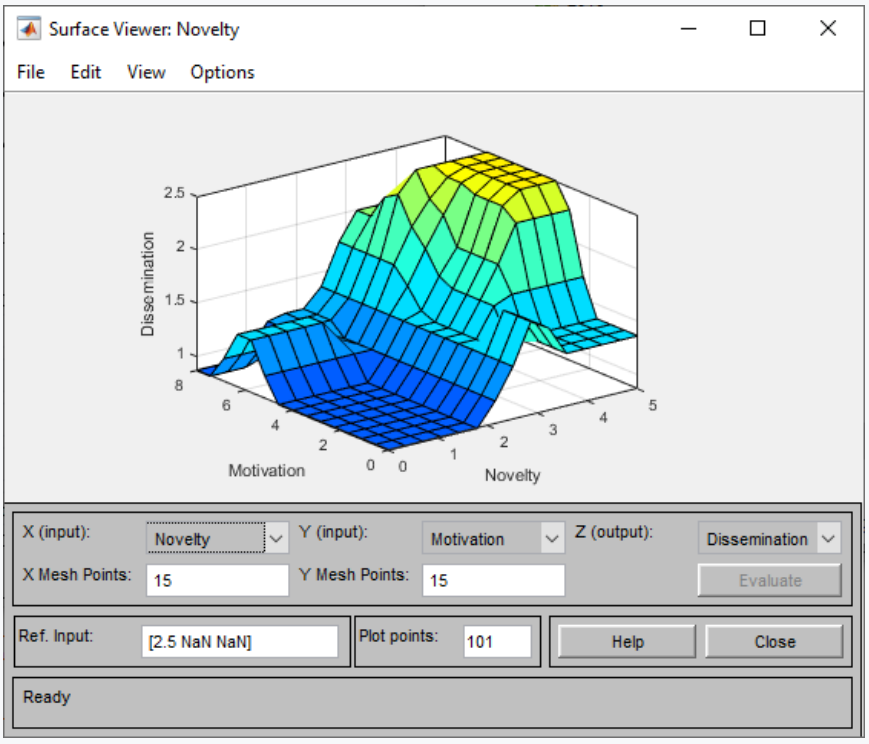
```
1 1 6, 1 (1) : 1
1 1 7, 1 (1) : 1
1 2 1, 1 (1) : 1
1 2 2, 1 (1) : 1
1 2 3, 1 (1) : 1
1 2 4, 1 (1) : 1
1 2 5, 1 (1) : 1
1 2 6, 1 (1) : 1
1 2 7, 1 (1) : 1
1 3 1, 1 (1) : 1
1 3 2, 1 (1) : 1
1 3 3, 1 (1) : 1
1 3 4, 1 (1) : 1
1 3 5, 1 (0.5) : 1
1 3 5, 2 (0.5) : 1
1 3 6, 2 (1) : 1
1 3 7, 1 (0.5) : 1
1 3 7, 2 (0.5) : 1
1 4 1, 1 (0.5) : 1
1 4 1, 2 (0.5) : 1
1 4 2, 2 (1) : 1
1 4 3, 2 (1) : 1
1 4 4, 2 (1) : 1
1 4 5, 2 (1) : 1
1 4 6, 1 (0.5) : 1
1 4 6, 2 (0.5) : 1
1 4 7, 1 (1) : 1
2 1 1, 1 (1) : 1
2 1 2, 1 (1) : 1
2 1 3, 1 (1) : 1
2 1 4, 1 (1) : 1
2 1 5, 1 (1) : 1
2 1 6, 2 (1) : 1
2 1 7, 1 (1) : 1
2 2 1, 1 (1) : 1
2 2 2, 1 (1) : 1
2 2 3, 1 (1) : 1
2 2 4, 1 (1) : 1
2 2 5, 1 (1) : 1
2 2 6, 1 (1) : 1
2 2 7, 1 (1) : 1
2 3 1, 2 (1) : 1
2 3 2, 1 (1) : 1
2 3 3, 1 (1) : 1
```

```
2 3 4, 1 (1) : 1
2 3 5, 1 (0.5) : 1
2 3 5, 2 (0.5) : 1
2 3 6, 2 (1) : 1
2 3 7, 2 (1) : 1
2 4 1, 1 (1) : 1
2 4 2, 1 (1) : 1
2 4 3, 2 (1) : 1
2 4 4, 2 (1) : 1
2 4 5, 2 (1) : 1
2 4 6, 2 (1) : 1
2 4 7, 2 (1) : 1
3 1 1, 1 (1) : 1
3 1 2, 1 (1) : 1
3 1 3, 1 (1) : 1
3 1 4, 1 (1) : 1
3 1 5, 1 (1) : 1
3 1 6, 1 (1) : 1
3 1 7, 1 (1) : 1
3 2 1, 1 (1) : 1
3 2 2, 1 (1) : 1
3 2 3, 1 (1) : 1
3 2 4, 1 (1) : 1
3 2 5, 1 (1) : 1
3 2 6, 1 (1) : 1
3 2 7, 1 (0.5) : 1
3 2 7, 2 (0.5) : 1
3 3 1, 2 (1) : 1
3 3 2, 2 (1) : 1
3 3 3, 2 (1) : 1
3 3 4, 2 (1) : 1
3 3 5, 2 (1) : 1
3 3 6, 2 (1) : 1
3 3 7, 2 (1) : 1
3 4 1, 2 (1) : 1
3 4 2, 2 (1) : 1
3 4 3, 3 (1) : 1
3 4 4, 3 (1) : 1
3 4 5, 3 (1) : 1
3 4 6, 3 (1) : 1
3 4 7, 2 (1) : 1
4 2 7, 1 (1) : 1
```

The final surface presents, that the rules cover all parts of the space:







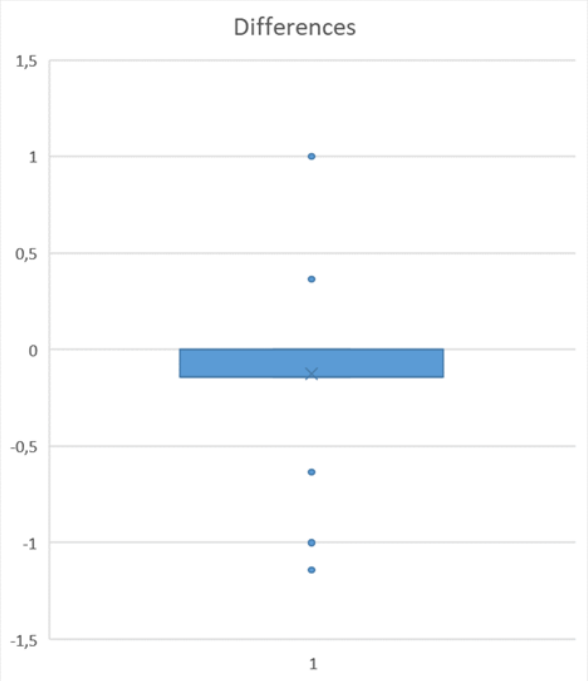
So we can proceed to testing the expert system.

📖 4.1.9

Expert system testing

The accuracy of the assessment created by the ES was verified on a sample of family firms from the industrial area. This testing identified small differences between the expert system and the sample data, which led to the conclusion that the expert estimate is slightly more pessimistic and the reality is more optimistic, as we can see in the evaluation of the deviations of the ES evaluation result compared to the company evaluation in Figure 3. Therefore, in the next stage, it will be appropriate to check the correctness of the individual if-then rules and deal with the reason for the occurrence of outliers. The reason may be an incorrect rule, but it may also indicate the need to refine the estimate by applying other parameters.

Below we see the deviations of the evaluation of the expert system compared to the evaluation of the innovativeness of the company:



We can see that the opinions of experts are slightly pessimistic. The evaluation identified only small deviations, which clearly proves that the chosen path as well as the identified parameters and their relationships are correctly set and the model is usable and applicable for evaluating the innovativeness of not only family businesses.

It will also be interesting to focus on outliers and check whether the evaluation of the companies was justified. Both in a positive and negative direction.

4.1.10

What do you see as important in creating a fuzzy expert system?

- Correct identification of input parameters.
- Correct determination of the values of individual input parameters.
- Correct determination of the values of the output quantity.
- Correct construction of IF-THEN rules.
- Sufficient coverage of the entire value space with rules.
- Expert System Builder Hourly Wage.

Decision Systems Modelling

Chapter 5

5.1 Generalized algorithm for modelling

📖 5.1.1

In decision system modeling, it's crucial to simulate human-like decision-making processes and to understand how knowledge can be acquired from user interactions. Knowledge acquisition in these systems refers to gathering insights based on user responses or behaviors while retrieving information. This is particularly important in the context of simulating decision-making systems on computers, as these systems aim to mimic human mental activities such as planning, coordinating, and making decisions. To perform these functions, a decision system must be able to simulate human knowledge acquisition and decision-making accurately.

For a system to substitute or support human activities, it needs strong communication abilities and must provide tools to assist users in specific tasks. Additionally, the system should be designed to apply heuristics—rules of thumb that can make complex decision-making simpler—alongside structured knowledge. It should also explain its decision-making process and derivations to the user, fostering transparency and enhancing trust in the system's choices. This explanation process allows users to follow the reasoning of the system and understand the steps taken to reach conclusions.

Finally, a decision system should seamlessly incorporate new information or "knowledge increments" into its existing database without disrupting prior data. This feature is essential for adapting to changing scenarios and staying relevant. By integrating new knowledge efficiently, the system remains up-to-date and can support users effectively, making it a dynamic decision-support tool.

📝 5.1.2

In decision systems, what does knowledge acquisition primarily involve?

- Gathering insights based on user responses
- Processing only stored data
- Focusing exclusively on analytic data
- Generating new data without user input

📝 5.1.3

Which feature is essential for a decision system to support human tasks?

- Ability to communicate well with users
- Simple integration of new knowledge

- Strict reliance on analytic data
- Limiting explanations of derivations

📖 5.1.4

Effective decision systems have several critical features that allow them to function similarly to human decision-makers.

One essential feature is the application of **heuristics-methods based on experience** or intuitive judgment rather than rigid calculations. Heuristics enable decision systems to handle complex scenarios with a degree of flexibility, allowing them to operate efficiently even when detailed data or a full algorithm is not available. This is vital for creating systems that can address real-world, uncertain scenarios.

Another significant feature is **the ability to explain** both **the knowledge** used **and the reasoning process** in deriving conclusions. By providing explanations, the system can foster user trust and understanding, which are crucial for decision support systems that often deal with high-stakes scenarios. For instance, if a healthcare decision system recommends a treatment plan, understanding the reasoning behind the recommendation is vital for both doctors and patients. This transparency also makes it easier to identify and correct errors.

Lastly, the system must be able **to integrate new knowledge incrementally**. This ensures that it can keep pace with changing information and adapt to new findings or rules. In an ever-evolving field like finance, for example, decision systems must update constantly to reflect new regulatory standards or market trends. Such flexibility makes these systems adaptable and more valuable over time.

📝 5.1.5

Which feature allows decision systems to make choices without full information?

- Applying heuristics
- Using strict algorithms
- Ignoring new information
- Providing rigid explanations

5.1.6

Why is it important for decision systems to explain their reasoning?

- To foster user trust
- To help users understand decisions
- To avoid system errors
- To eliminate complex data analysis

5.1.7

Simulating decision-making processes in a system involves reproducing the cognitive and abstract reasoning that humans use. Unlike purely analytical systems, decision-making simulations rely not only on data but also on experiential knowledge and cognitive models. These simulations are capable of adapting to different scenarios and judgments by mimicking the flexible thought processes unique to human intelligence. For instance, in diagnosing a problem, a human expert considers both objective and subjective information, a balance that decision systems must strive to achieve.

Furthermore, decision-making is complex due to the variety of methods available, depending on the number of individuals involved in making a decision. For instance, a single person making a decision might focus solely on specific data, while group decision-making can incorporate diverse perspectives, thus requiring additional layers of analysis. Decision systems aim to adapt to these nuances, often by allowing for different decision-making protocols based on the context.

Simulating decision-making also requires a method to handle external information. In practice, decision systems often rely on external databases to provide context or additional insights, ensuring that decisions consider broader information beyond the immediate dataset. This external data could include market trends, weather patterns, or legal guidelines, helping the system make more robust decisions.

5.1.8

In a decision-making system, what unique process does simulation attempt to replicate?

- Cognitive and abstract reasoning
- Only analytical data processing
- External data manipulation only
- User preferences exclusively

5.1.9

Why is external information important in decision-making simulations?

- It enhances decision relevance
- It ensures broader context
- It limits the need for algorithms
- It reduces system flexibility

5.1.10

The decision-making process can be broken down into three core phases: information acquisition, planning, and selection.

- The first phase, **information acquisition**, involves gathering and analyzing relevant data. This phase is crucial because it forms the foundation upon which the decision will be built. In information systems, this often involves drawing on both internal and external databases to collect accurate and timely information, which helps to contextualize the decision.
- **Planning** is the next phase, where alternatives are considered and evaluated. This involves assessing different potential outcomes and weighing their pros and cons. For instance, a company planning a new product launch might evaluate multiple strategies and decide on the one with the most potential. The planning phase is where creativity and analysis intersect, as decision-makers think critically about the various options and potential results.
- Finally, the **selection** phase involves choosing the best option among the alternatives. Here, the system's ability to balance conflicting factors, such as cost and quality, plays a significant role. This phase often relies on predefined rules or criteria within the system to ensure that decisions align with organizational objectives. After selection, the decision-making process can move to implementation, where the decision is put into action.

5.1.11

A decision-making process can be defined as an organic unity of three phases in the order:

- selection (variant selection);
- information (knowledge acquisition);
- planning (considering alternatives);

5.1.12

What does the planning phase in decision-making involve?

- Evaluating different alternatives
- Only collecting data
- Final selection of an option
- Ignoring creative input

5.1.13

In uncertain environments, decision systems must use specialized algorithms to handle incomplete or ambiguous information. A generalized algorithm for decision system modeling under uncertainty provides a structured way to make decisions even when the data is not fully clear. This algorithm involves assessing probabilities and using heuristics to manage uncertainty effectively. By accounting for unknowns, the system can generate recommendations that are reliable even in less predictable situations.

One example of such an algorithm is presented in [Klimes 2011], which outlines methods for decision modeling in complex environments. The algorithm allows for adapting to changing data inputs and making decisions with flexible criteria. This is especially useful in industries like finance or healthcare, where sudden shifts in data may occur frequently, requiring systems that can respond dynamically.

Through this algorithmic approach, decision systems can perform tasks traditionally reliant on human judgment, like forecasting trends or diagnosing issues. The ability to work within uncertain parameters makes these systems valuable for businesses and organizations that deal with complex and evolving data sets.

5.1.14

What is the purpose of a generalized algorithm in uncertain environments?

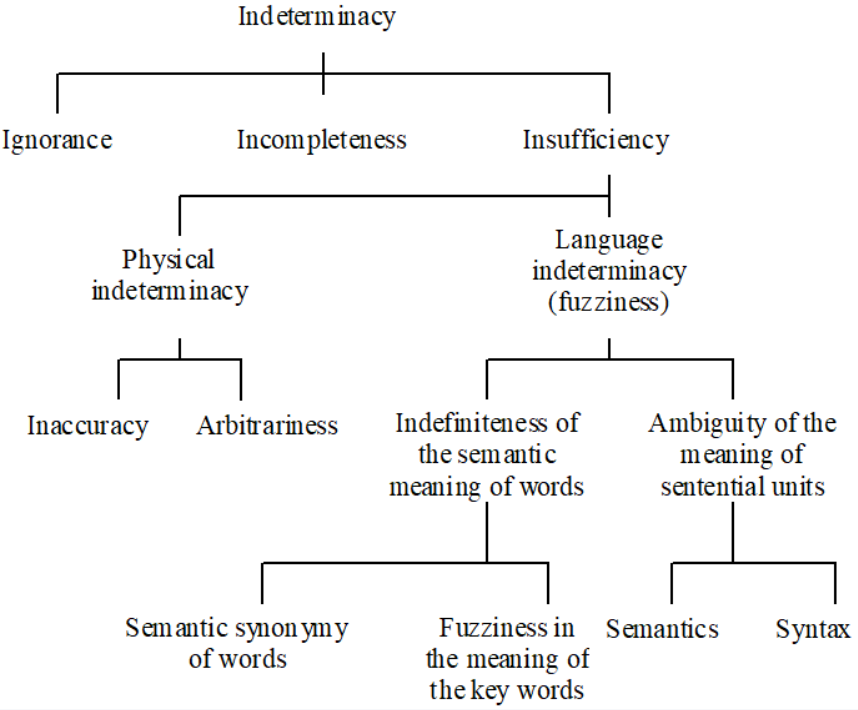
- To handle incomplete or ambiguous information
- To avoid making decisions
- To simplify the decision process completely
- To disregard unknown factors

5.2 Uncertainty in decision systems

🕮 5.2.1

In decision-making processes, one key feature is the frequent use of indeterminate and non-metric information. Unlike quantitative data, indeterminate information lacks clear, measurable values and instead is influenced by subjective human input, such as experience and opinion. This subjectivity introduces uncertainty and complexity into decision-making, as it makes the process harder to quantify or define with precision. When humans provide the input for these systems, they rely on personal judgment, which can vary widely across individuals and situations.

Indeterminacy in decision-making has a structure that can be managed by using specific tools designed to handle incomplete or imprecise information. The basic structure of incompleteness in the area of information (we are primarily interested in this area) can be depicted by the following scheme:



This structure, particularly within information systems, allows decision-making processes to address varying degrees of incompleteness systematically. For instance, a decision-making framework might employ statistical models or specialized algorithms to manage the gaps in knowledge due to subjective input. By structuring and analyzing these gaps, decision-makers can better manage the inherent uncertainty involved.

This form of uncertainty can stem from two main sources: physical indeterminacy and linguistic limitations. Physical indeterminacy arises from measurement inaccuracies, where it's challenging to predict certain quantities accurately due to their inherent randomness. Meanwhile, linguistic limitations refer to the constraints

of natural language, where descriptions are often too vague or broad to capture all nuances of a decision-making process. Since words have varying interpretations, the information conveyed can be ambiguous, adding another layer of complexity to decision-making.

5.2.2

One of the characteristic features of decision-making processes is the fact that

- they often work with indeterminate and non-metric information...
- they often work with intercontinental and non-deterministic information...
- they often work with interconnections and non-commercial information...

5.2.3

Which factor often causes indeterminate information in decision-making?

- Human experience and opinion
- Precise measurement tools
- Machine learning algorithms
- Exact mathematical models

5.2.4

Physical indeterminacy in decision-making is primarily due to

- Measurement inaccuracies
- Unpredictable occurrences
- Machine precision
- Consistent data inputs

5.2.5

Indeterminacy in decision-making often manifests due to insufficient information, which can arise from external and linguistic factors. External factors, or physical indeterminacy, stem from measurement errors and the unpredictable nature of some elements, making it impossible to achieve absolute accuracy. For example, in scientific experiments or economic forecasts, slight variations in measurements can significantly alter the outcome, introducing uncertainty into the decision-making

process. The randomness in external factors thus contributes to the complexity of making informed decisions.

Another prominent source of indeterminacy is the language used by humans to describe decisions. Natural language, while flexible and expressive, often lacks the precision needed for clear, definitive communication. When humans input information into a decision-making system, they are bound by limitations in language, which cannot fully capture every detail or possibility within a finite timeframe. This finite use of words creates gaps in communication, as words can have multiple meanings, and nuances might be lost, leading to further ambiguity.

Such linguistic indeterminacies affect the clarity of information used for decision-making. A human's choice of words may unintentionally introduce bias or fuzziness, creating a semantic field where meanings can overlap or contradict. This variance in interpretation can make it difficult for a system to process information accurately, underscoring the need for refined methods to interpret linguistic input. Addressing these challenges requires specialized tools that can work with flexible or "fuzzy" information.

5.2.6

What is one main reason natural language introduces indeterminacy?

- Natural language lacks numerical precision
- Words have a finite number of interpretations
- Language is consistent across contexts
- Exact mathematical terms are always used

5.2.7

Which factors contribute to linguistic indeterminacy?

- Overlapping meanings of words
- Semantic synonymy
- Consistent terminology
- Precise definitions

5.2.8

The fuzziness in language significantly impacts decision-making, as traditional mathematics and exact sciences often struggle to handle linguistic uncertainty effectively. Words and phrases can hold multiple meanings or degrees of

interpretation, which introduces ambiguity. For example, terms like "high risk" or "significant impact" may carry different connotations depending on the context, making them difficult to quantify precisely. This fuzziness has traditionally limited the application of exact sciences in linguistically-defined scenarios, as traditional methods require precise data.

The development of fuzzy mathematics, however, has transformed the ability of decision systems to work with this ambiguity. Fuzzy mathematics is designed to model situations that involve vagueness and imprecise information, allowing it to process linguistic descriptions in a structured manner. For instance, a fuzzy logic system can assign degrees of truth to statements rather than using binary true/false outputs. This flexibility allows decision-making systems to accommodate and work effectively with non-metric data, bridging the gap between linguistic and numerical information.

By applying fuzzy mathematics, decision systems can better handle situations where human language plays a central role in the input. This approach has become essential in fields like artificial intelligence, where systems need to interpret and act on instructions or descriptions provided by humans. Through fuzzy logic, systems are now capable of interpreting ambiguous information, allowing them to make decisions based on a range of inputs rather than relying on fixed numerical values alone.

5.2.9

Why was fuzzy mathematics developed?

- To model situations with vague information
- To increase measurement precision
- To eliminate linguistic input in systems
- To reduce decision-making speed

5.2.10

Fuzzy mathematics is useful in decision-making because it

- works with linguistic uncertainty
- allows non-metric data interpretation
- only uses numerical data
- eliminates ambiguity

5.3 Decision process model

🕮 5.3.1

he fundamental elements of a decision system. Each set plays a unique role in how the system processes information and makes decisions.

1. **S – Situations**: This set represents all the potential states or conditions that the system might encounter. Each situation within **S** provides context for decision-making and could represent anything from market conditions for a business to health metrics for a medical system. Understanding the current situation is crucial, as it defines the parameters for which decisions need to be made. A decision system uses this situational awareness to evaluate which actions will lead to optimal outcomes.
2. **D – Possible Solutions**: The set **D** includes every potential solution or decision the system could make in response to any situation from **S**. These solutions are the possible actions or responses the system can take, and each solution has unique implications for the target goals. For example, a business decision-making system might consider pricing strategies, marketing channels, or customer service options as potential solutions. The system evaluates each option within **D** to determine which best aligns with the overall objectives defined in **G**.
3. **G – Targets/Goals**: The target set **G** includes all acceptable outcomes or goals the system aims to achieve. These targets provide a measure against which the success of each solution is evaluated. In a healthcare system, for instance, **G** might include outcomes such as patient recovery or symptom reduction. In a decision system, every decision made from **D** is checked for how well it meets one or more goals from **G**.
4. **F – Degrees of Existence (Probabilities)**: This set represents the likelihood or existence probability of each object or element within the decision-making environment. Probabilities allow the system to handle uncertainty, weighing each possible solution by how likely it is to yield a beneficial outcome given the current situation. By incorporating probabilities, the system can prioritize actions based on their probable effectiveness.
5. **K – Evaluations**: The evaluations set **K** assigns a score or rating to each solution, allowing the system to assess each option's performance or suitability. Evaluations might consider factors like efficiency, cost, or satisfaction, depending on the system's objectives. For example, in a recommendation system, **K** could include user satisfaction ratings that help refine future recommendations.
6. **T – Time Interval**: Finally, **T** represents the time frame within which decisions and evaluations occur. Time is a critical factor in decision systems, as outcomes often depend on timely responses. For dynamic or real-time systems, decisions may need to be reassessed periodically, with **T** helping the system know when to evaluate and update its decisions based on new information or changing circumstances.

Together, these elements create a framework for decision systems, enabling them to analyze situations, evaluate options, consider uncertainties, and ultimately select solutions that best align with predefined goals within a specific timeframe.

5.3.2

Which set represents the possible actions or responses the system can take in response to various situations?

- D – Possible Solutions
- S – Situations
- G – Targets/Goals
- K – Evaluations

5.3.3

Which of the following sets are used to determine the effectiveness and appropriateness of solutions in a decision-making system?

- K – Evaluations
- G – Targets/Goals
- T – Time Interval
- F – Degrees of Existence

5.3.4

Despite the fact that the decision-making process involves numerous indeterminacies, its structure can be defined relatively well. We know, the elements of decision-making process can be divided into the following groups:

- S – a set of situations,
- D – a set of all possible solutions,
- G – a set of all targets (admissible) for further functioning of a given system,
- F – a set of all degrees of existence (probabilities) of a given object,
- K – a set of all evaluations for a given solution,
- T – time interval.

The decision-making process itself is represented by various mappings among these sets. It particularly concerns the following mappings:

1. the process of information completion about a given situation and its evaluation, i.e. selection of only the information which has importance for the final solution:

$$M_1 : S \times T \times F \rightarrow S \times T \times F$$

2. the process of creating a set of admissible solutions, which consists of two partial processes

$$M_2 = M_{22} \circ M_{21}$$

where:

- M_{21} – formulating management targets based on the description of a given situation,
- M_{22} – formulating admissible solutions:

$$M_{21}: S \times T \times F \rightarrow G \times S \times T \times F$$

$$M_{22}: G \times S \times T \times F \rightarrow D \times S \times T \times F$$

3. the process of modelling effects of admissible solutions

$$M_3 : D \times S \times T \times F \rightarrow D \times S \times T \times (S \times T)^* \times F$$

where $(S \times T)^*$ defines the set of all chains over $(S \times T)$,

(where each admissible solution is allocated with a set of situations including their time courses, which arise from the given decision.

4. the process of acceptance of the solution itself, which consists of two partial processes

$$M_4 = M_{42} \circ M_{41}$$

where:

- M_{41} – evaluating the behaviour of effects of admissible solutions,
- M_{42} – selection of the best variants:

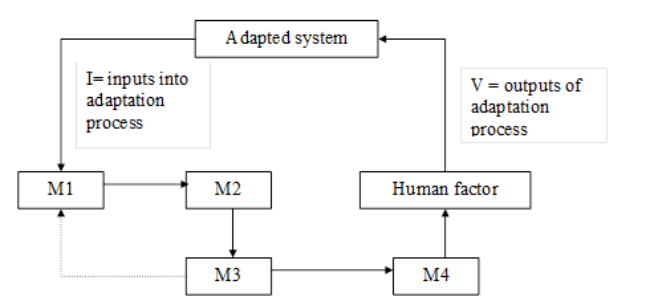
$$M_{41} : D \times S \times T \times (S \times T)^* \times F \rightarrow D \times K \times T \times F$$

$$M_{42} : D \times K \times T \times \rightarrow D \times T$$

The whole decision-making process is created through the gradual composition of these partial processes

$$M = M_4 \circ M_3 \circ M_2 \circ M_1$$

As we can depict in the following diagram



Let us note that the realization of particular processes M_1 – M_4 can be ensured by means of so-called fuzzy algorithms using the results of fuzzy sets theory.

Particular processes M_1 – M_4 differ one from another by the character of input and output quantities as well as by other relations being performed within the process.

5.3.5

The decision-making process can be divided into the following groups:

- S – a set of situations,
- D – a set of all possible solutions,
- G – a set of all targets (admissible) for further functioning of a given system,
- F – a set of all degrees of existence (probabilities) of a given object,
- K – a set of all evaluations for a given solution,
- T – time interval,
- A – a set of alfanumeric symbols representing the possible solutions.
- B – a set of bad decisions,
- C – a set of certain coins needed to generate the values of a random variables,

5.3.6

The set of all possible _____ is known as D, while the set of all _____ for a particular solution is represented by K. The set G includes all acceptable _____ for the system.

- evaluations
- solutions
- goals

5.3.7

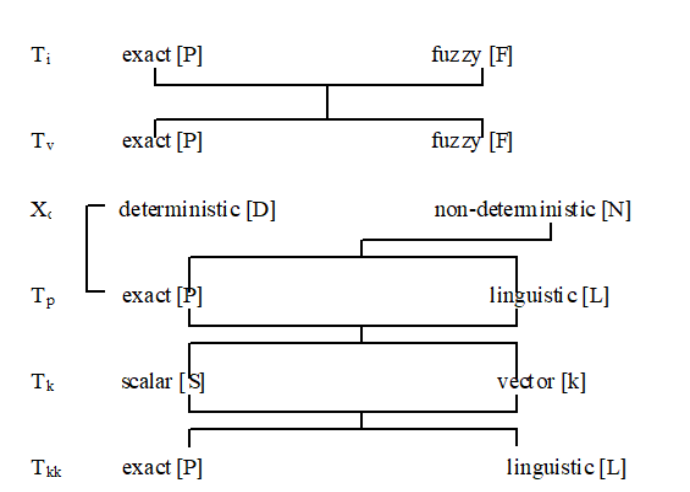
Generally speaking, each process can be featured by the following sextuplet

$$(T_i, T_v, X_{ch}, T_p, T_k, T_{kk})$$

where

- T_i – type of inputs,
- T_v – type of outputs,
- X_{ch} - character of outputs,
- T_p – type of indeterminacy,
- T_k – type of selection criterion,
- T_{kk} – type of particular elements of the criterion.

The particular types may take the values given in the following diagram:



When creating the simulation system, we assumed that it would not be necessary to deal with all processes $M_1 ... M_{41}$, M_{42} because some of them are evident.

We can presume that the process of information completion about a given situation and its evaluation M_1 is a part of the acquisition of the input situation evaluation and it is not necessary to explain it in more details.

The key processes in the decision-making process can be considered processes

$$M_{22}, M_3, M_{41}, M_{42}.$$

📝 5.3.8

The key processes in the decision-making process can be considered processes:

- M22, M3, M41, M42.
- MM, MN, MO, MI, MP, MPS.
- A, B, C, D, E, F, G, H, I, J.
- M1, S2, T3, K4, U1, U2, U3.

5.4 Model of web system adaptation

🕮 5.4.1

Project: Web system adaptation proposal

Adaptive web systems are designed to tailor web content based on the behavior and characteristics of individual users. By monitoring user interactions, these systems identify patterns and preferences, offering information that aligns with each user's unique needs. The primary motivation for creating adaptive systems lies in the diversity among users—people have different abilities, preferences, and goals when accessing online content. Adaptive web systems seek to bridge these differences by adapting content, user interfaces, and information structure based on observed user behavior.

In an adaptive web system, the user interface may change depending on user needs. For example, a student studying for an exam might be presented with condensed summaries, while a researcher could see more in-depth analyses and detailed reports. The system constantly evaluates and updates the stored user information, making it possible to personalize content dynamically. However, this level of adaptation means users cannot be anonymous—each user's data must be continuously collected and assessed to provide effective personalization.

The goal of an adaptive system is to increase user satisfaction and efficiency by delivering content that meets individual needs. This approach improves information relevance and accessibility, providing users with a streamlined experience that

matches their goals. Ultimately, adaptive systems aim to reduce information overload and increase comprehension, especially for users who may benefit from tailored content recommendations.

5.4.2

Adaptive web systems personalize _____ based on the behavior and characteristics of each _____, ensuring content matches their individual _____.

- needs
- user
- content

5.4.3

What is the primary purpose of an adaptive web system?

- To tailor information to each user's needs
- To allow users to remain anonymous
- To ensure content is identical for all users
- To limit access to information

5.4.4

When an adaptive system selects which information to display, it considers various factors. These factors are crucial for determining what information will be most helpful and relevant to the user. Key factors include data volume, information content, and the importance of the information. Additionally, adaptive systems account for the availability, credibility, cost, and acquisition time of information. For example, in a rapidly changing field like news reporting, timely and credible information is prioritized over less relevant or outdated data.

Another important consideration is the amount of space that information takes up on a system's resources, such as disk storage. For instance, large multimedia files might be stored less frequently than text-based files due to space constraints. The cost and time associated with acquiring information also play significant roles, as some data might require extensive time or resources to collect. An efficient system will balance these factors to deliver content in a cost-effective and timely manner.

In essence, the decision-making process for information selection involves a nuanced evaluation of these factors. The system's ultimate goal is to identify and present information that balances quality, credibility, and accessibility, ensuring that

users receive content that aligns with their preferences without overloading system resources.

5.4.5

Which factors are considered in an adaptive web system’s decision-making process for selecting information?

- Credibility
- Information content
- User's privacy settings
- Size of the user’s device

5.4.6

Why is the cost of information acquisition an important factor in adaptive systems?

- To manage system resources efficiently
- To limit user access to information
- To avoid storing any data
- To increase the amount of displayed data

5.4.7

The output of an adaptive web system is the provision of tailored solutions represented by certain characteristics, with "suitability of provided information" being a critical focus. Suitability is judged on factors like the relevance of the content, ease of understanding, and frequency of user interactions with the content. For example, if a topic is frequently accessed, it indicates high relevance and may be featured more prominently for users with similar interests.

The system can also recommend information sources based on user interactions. For instance, if a user frequently searches for medical content, the system may suggest credible health databases as additional resources. Comprehensibility is essential; even if the content is accurate, it must be accessible to users of varying expertise levels. In cases where users need additional resources, the system can suggest other sources, enhancing the overall user experience.

Finally, adaptive systems can simulate the effects of decisions, helping developers refine the system to improve accuracy and relevance. By analyzing user feedback, the system can adjust future recommendations, creating a loop of continuous

improvement. This process makes the system more intuitive and aligned with user needs over time.

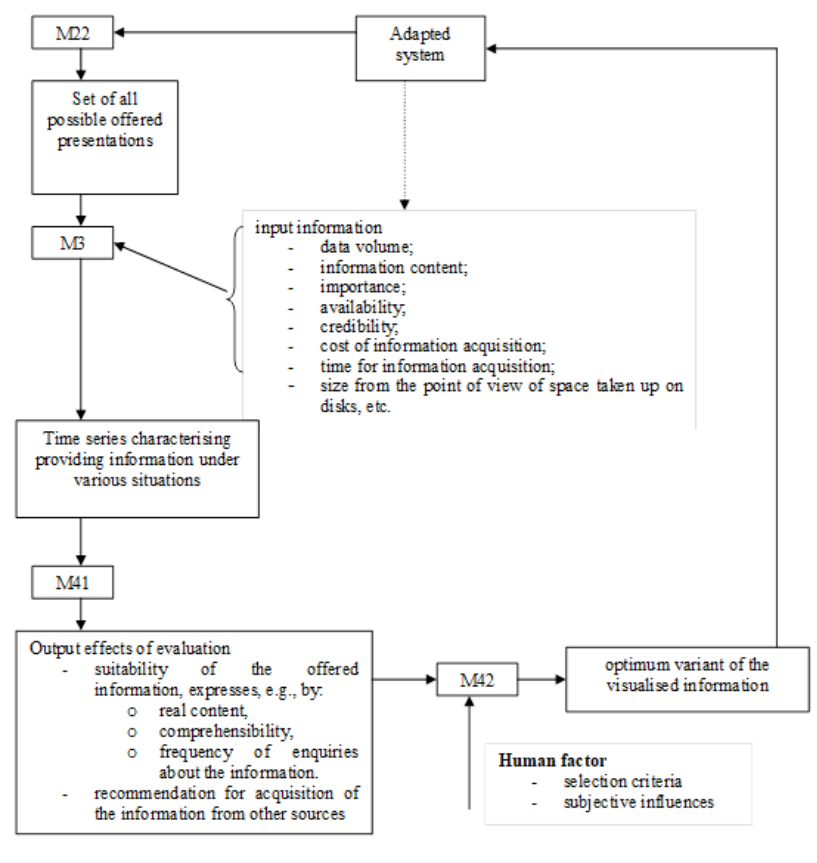
5.4.8

What are key characteristics of suitable information in an adaptive system?

- Relevance to user needs
- Comprehensibility
- The highest cost
- Storage in multiple languages

5.4.9

Based on this specification, we can define the following general structure of the decision-making system solving a given problem:



5.4.10

What is the basic motivation to create the adaptive web system?

- The variability of individual users.
- The number of different parameters.
- The insufficient knowledge of the problem.

5.5 Model of optimalisation

5.5.1

Project: Optimalisation

First, let us consider the M_{42} process. From the classification point of view, as depicted in the process types figure, the process of selecting the optimum variant belongs to the following two key categories:

- I. category = (P, P, D, P, V, P)
- II. category = (F, F, N, L, V, L)

The two categories correspond to the fact that during the selection of the optimum variant, the input quantities are entered either exactly or verbally, the output quantity is either exact (detailed analysis of the visualization variant) or on the contrary as a verbally described suitability of the given mapping, and a vector criterion entering the selection of mapping can also be described either defined exactly or set verbally.

It is evident that the realisation of both processes M_{42}^{I} and M_{42}^{II} is quite different. While in the case of M_{42}^{I} process, it concerns a classic vector optimisation, in the case of M_{42}^{II} process, the situation is quite different.

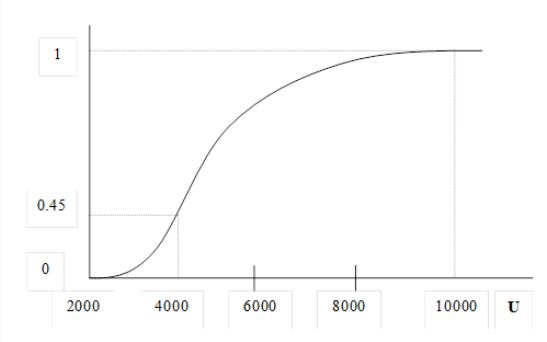
We stem from the fact that the inputs of the decision-making process M_{42}, as well as the decision-making algorithm itself, can be described uncertainly, primarily by linguistic terms characterising values of specific quantities or by relations between specific quantities. However, to make the performance of such inputs and algorithms computerised, it is necessary to use a suitable mathematical apparatus One of the possibilities for how to perform such quantities is the fuzzy set theory. We will mention here some descriptions from this theory that are possible to be used in a decision-making process.

Let U be a set of objects, which are concerned in our decision-making process (e.g. U is a time interval, or an interval representing average costs to acquire information

expressed, e.g., by the amount of database passes, etc.). The fuzzy set in *U* will be called the mapping

$$A : U \rightarrow [0,1]$$

where quantity $A(x),\ x \in U,$ is called by the membership level of the element *x* in *A*. As for each function, we can represent *A* by the guarantee of the function. For example, if *U= [2000, 10 000]* is the universe representing the quantity of information, then *A* in *U* is defined by the diagram in the next figure:



represents the verbal term: *A = HIGH INFORMATION LEVEL*.

The fact that *A* is a fuzzy set in *U* will be marked as $A \subseteq U$. Furthermore, the term "fuzzy relation" is important for our purposes. If U_1 *and* U_2 are two universes, then fuzzy relation is a fuzzy set in their Cartesian product, i.e. $R \subseteq U_1 \times U_2$. If e.g. $U_1 = U_2 = U$ in the previous example, then we can define the fuzzy relation $R = NEARLY\ EQUAL\ to \subseteq U \times U$ by means of a functional prescription

$$R(x, y) = e^{-|x-y|}, x, y \in [2000, 10000]$$

Similarly to classical sets, we can define analogue operations within the class of fuzzy sets. Particularly, if *U* is a universe, $A, B \subseteq U$, then we define:

- $(A \cup B)(x) = \max\{A(x), B(x)\}$
- $(A \cap B)(x) = \min\{A(x), B(x)\}$
- $\neg A(x) = 1 - A(x)$
- $(A \times B)(x, y) = \min\{A(x), B(y)\}$

For our next objectives, it is important to introduce the term linguistic variable, i.e. the variable χ represented by the following system

$$\chi = \langle X, \tau, M \rangle$$

where *X* is a domain of values, τ is a set of terms (i.e. specific words) and *M* is semantics, i.e. representation assigning a fuzzy set $M(t) \subseteq X$ to each term *t*.

If we consider the linguistic variable

$$\chi = SIZES.$$

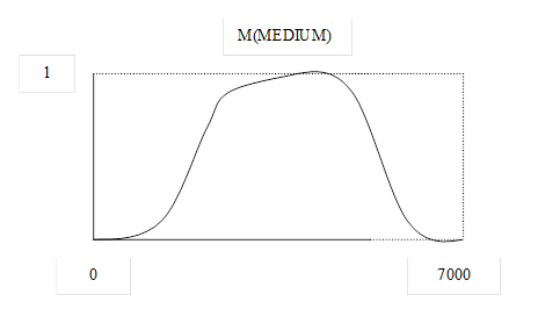
The set of terms *t* of this variable is created e.g.by words

$$\tau = \{HIGH, LOW, MEDIUM, VERY\ HIGH, ...\}$$

Then we can define as a domain of values, e.g., interval

```
X = <0, 700>
```

(e.g. *t* suitability of the offered information) and function *M* for particular terms can be defined e.g. in the following diagrams and relations:



and furthermore

$$M\ (VERY\ t)\ (x) = [M(t)\ (x)]^2\ ;\ x \in X,$$
$$M\ (NOT\ t)\ (x) = 1 - M(t)\ (x),$$
$$M\ (t_1\ AND\ t_2)\ (x) = \min\ \{M(t_1)\ (x), M(t_2)\ (x)\},$$
$$M\ (t_1\ OR\ t_2)\ (x) = \max\ \{M\ (t_1)\ (x), M\ (t_2)\ (x)\}.$$

The,n for example, the suitability level of the offered information

```
x = 3000 t
```

corresponds to the verbal expression

```
t = NOT VERY HIGH AND NOT VERY LOW
```

with the membership degree

M(t)(x) = min {M (NOT VERY HIGH) (x), M(NOT LOW) (x)}= min {1-M (VERY HIGH) (x), 1-M (LOW) (x)} = min {1-(M(HIGH)(x))^2, 1-M(LOW)(x)} = min {1-0.25^2, 1-0.55} = min {0.9375, 0.45} = 0.45

i.e. only a half.

📖 5.5.2

Using linguistic variables, we can set up fuzzy algorithms of certain processes.

If the input quantities of a given system are $x=(x_1..x_n)$ and the output ones $y=(y_1..y_m)$, then the fuzzy algorithm means the expression:

If $\varphi(x_1..x_n)$, **then** $s(y_1..y_m)$, **or** $y(y_1..y_m)$,

- where $\varphi(x_1..x_n)$, $s(y_1..y_m)$ $y(y_1..y_m)$ are linguistic expressions concerning the individual mentioned quantities.

For example, in the process M_{42}^{II} we consider the following situation. One of the partial decision-making rules in this process can concern e.g. the relations among the importance (e), availability (v), unit costs for information acquisition (j), and the selection of the given mapping (t).

Let us suppose that $e \in \langle a_1, b_1 \rangle, v \in \langle 0,100\% \rangle$. Then the verbal expression of one of the decision-making rules can be as follows:

```
R1: if e = MEDIUM, v = HIGH, j = MEDIUM, then t = HIGH
SUITABILITY
```

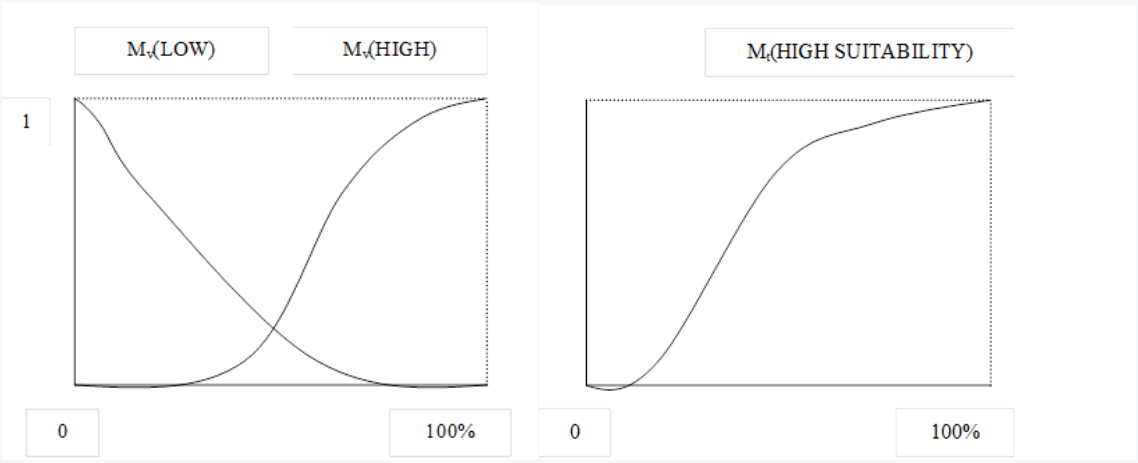
or

```
R2: if e = HIGH, v = HIGH, j = HIGH, then t = MEDIUM
SUITABILITY.
```

We then deal with the relation among 4 linguistic variables

```
E = <{LOW, MEDIUM, HIGH, VERY, AND, NOT, OR}, <a1,b1>,
Me>
V = <{LOW, MEDIUM, HIGH, VERY, AND, NOT, OR}, <0, 100 %>,
Mv>
J = <{LOW, MEDIUM, HIGH, VERY, AND, NOT, OR}, <a2,b2>,
M1j>
T = <{LOW SUITABILITY, MEDIUM SUITABILITY, HIGH SUITABILITY,
VERY, AND, NOT, OR}, <0,100%>, Mt>,
```

some fuzzy sets of which can be, e.g., as follows:



Generally, we can say that we possess such rules

```
R1: if e = A1, v = B1, j = C1, then t = D1
Rk: if e = Ak, v = Bk, j = Ck, then t = Dk.
```

In case we have a specified input vector (e,v,j), we can determine the corresponding value with a corresponding mapping t as follows.

Let d_i be an x-coordinate, the gravity center of the surface lying above the graph of function D_i, i.e.

$$d_i = (\int x.D_i(x)dx) / \int D_i(x)dx);\ i=1,\ \ldots,\ k \qquad x\in<0,100> \quad x\in<0,100>$$

Let furthermore

$$s_i = \min\ \{A_i(e),\ B_i(v),\ C_i(j)\} \in <0,1>$$

Then we can put

$$SUITABILITY(t) = \frac{\sum_{i=1}^{k} d_i.s_i}{\sum_{i=1}^{k} s_i}.100 \in \langle 0,100\% \rangle$$

📖 5.5.3

The set of rules R_i,...R_k can be preferably obtained by using expert assessments.

One of the problems connected with the application of fuzzy sets in process M_{42} is the problem of constructing the corresponding fuzzy sets $M(t)$, where t denotes the terms of the particular linguistic variables. We will show here several possible approaches to solving the problem.

Let us consider e.g. a linguistic variable V=INFORMATION AVAILABILITY and its term t = HIGH. We need to define functions M_v (HIGH) : [0,100] → [0,1].

(1) We have m available experts. For any value $x \in [0, 100]$, the experts answer the question of whether the value corresponds to the expression HIGH or not. Let n out of these experts confirm that it corresponds, then

$$M_v(HIGH)(x) = \frac{n}{m} \in [0,1]$$

(2) Le us assume again that we have m experts and only values $x = 0, 1, 2, ..., 100$ are being tested. (Note: We will use the principles, described in the part focused on Multi-criteria Decision Analysis as Saaty's method.) Each of the experts then defines values m_{ij} in such a way that:

- M_{ij} = 1, if he/she considers M_v (HIGH)(i) approximately equals to M_v (HIGH)(j)
- M_{ij} = 3, if he/she considers M_v (HIGH)(i) is a little bigger than M_v (HIGH)(j)
- M_{ij} = 5, if he/she considers M_v (HIGH)(i) is bigger than M_v (HIGH)(j)
- M_{ij} = 7, if he/she considers M_v (HIGH)(i) is a bit bigger than M_v (HIGH)(j)
- M_{ij} = 9, if he/she considers M_v (HIGH)(i) is much bigger than M_v (HIGH)(j)

If it was defined already m_{ij}, i < j,

it is put $m_{ji} = 1/ m_{ij}$.

If the maximum inherent number of the matrix $A = \|m_{ij}\|$, we can find the solution x = (x_1..x_{100}) of the matrix equation

$$(A - \alpha.E).X = 0$$

Then we put

$$M_v(HIGH)(i) = \frac{x_i}{\sum_{j=1}^{100} x_j}$$

The same situation is also in the case of process M_{41}, which can be also decomposed into two parts M_{41}^{I}, M_{41}^{II}. For the deterministic part of M_{41}, we can use classical methods for analyzing time series, which are usually available.

5.6 Application

📖 5.6.1

In decision-making modelling systems, the process is split into two primary stages: model creation and model application. Each stage has distinct objectives and requires different types of expertise. Model creation involves building the framework that will guide the decision-making process. This stage is complex because it requires defining fuzzy sets, which represent variables with uncertain or imprecise values, and establishing rules that will govern decisions. For example, creating a fuzzy set could involve defining what "high," "medium," or "low" risk means in a given context. Experts in the field are essential at this stage to provide insight and experience, ensuring that the model accurately reflects real-world conditions.

Once the model is built, the next step is to apply it to real data or scenarios. In this stage, input parameters are entered into the model, and computations are performed to arrive at outputs or decisions. Unlike the creation stage, applying the model is less analytically demanding, as it primarily involves straightforward mathematical operations, often handled by computer algorithms. These operations process inputs according to the rules and fuzzy sets defined during the creation phase.

The application stage also includes the model's recovery, which is the process of refining or adjusting the model based on new information or changing conditions. This ensures that the model remains relevant and accurate over time. Together, these stages allow a decision-making system to evolve, making it both adaptable and capable of handling complex, real-world situations.

📝 5.6.2

What is the primary focus of the model creation stage in a decision-making system?

- Defining fuzzy sets and establishing rules
- Processing input parameters
- Conducting simple mathematical operations
- Recovering data from past models

📝 5.6.3

In the decision-making modelling system, the _____ stage involves defining fuzzy _____, and establishing decision-making _____.

- creation
- sets
- rules

📖 5.6.4

Fuzzy sets are a fundamental concept in the model creation phase of decision-making systems. Unlike traditional sets where an element either belongs or does not belong, fuzzy sets allow elements to have degrees of membership. This approach is useful in decision-making because real-world situations are often uncertain or vague. For example, when assessing the risk of a financial investment, we may not be able to say with certainty whether it's "high risk" or "low risk" but rather somewhere in between. Fuzzy sets let us represent this ambiguity, assigning a degree of membership to categories like "high risk" or "medium risk."

The process of creating fuzzy sets requires a detailed understanding of the problem context. Experts work closely to define ranges and thresholds that reflect real-world variables. These definitions become the basis for the fuzzy sets used in decision-making rules. For instance, in a weather forecasting model, fuzzy sets might define temperature ranges as "cold," "warm," and "hot" with overlapping degrees of membership, providing a more nuanced way of interpreting temperature data.

By defining fuzzy sets, the model gains flexibility, allowing it to handle scenarios where data does not fit neatly into predefined categories. This adaptability is essential in complex decision-making systems, especially when they need to interpret data with uncertainty and variability.

📝 5.6.5

Which characteristics define fuzzy sets?

- Allow degrees of membership
- Represent uncertain data
- Require only binary membership
- Only apply to clear-cut data

📝 5.6.6

Why are fuzzy sets important in decision-making models?

- They handle uncertainty in data
- They eliminate the need for expert input
- They rely on clear boundaries
- They make models easier to create

📖 5.6.7

In model creation, decision-making rules are set to guide the system's responses to different scenarios. These rules determine how inputs are interpreted and what outputs should result. For example, in a quality control model, a rule might state that if a product's defect level is "high" (as defined by fuzzy sets), then the product should be rejected. Decision-making rules allow a model to make consistent and logical choices based on the conditions it encounters.

Creating decision-making rules is a collaborative process that involves input from experts who understand the practical aspects of the decision environment. They help ensure that rules align with real-world requirements and can adapt to various conditions. In this way, the system can make decisions in a manner that reflects human reasoning, even when faced with ambiguous or incomplete information.

Once established, these rules become the foundation for the model's decision-making capabilities. They allow the system to perform assessments and provide recommendations or actions, ensuring that responses are systematic and aligned with the defined goals of the system.

📝 5.6.8

Decision-making _____ provide structure for model responses based on specific _____, allowing the model to make consistent _____.

- inputs
- decisions
- rules

📝 5.6.9

What role do experts play in setting decision-making rules?

- Ensuring rules align with real-world needs
- Conducting mathematical calculations
- Performing repetitive data entry tasks
- Verifying all outputs

📖 5.6.10

After creating the model, the next step is to apply it using real data or scenarios. This process, known as model application, involves inputting parameters that the model

uses to generate outputs. The application phase focuses on processing the inputs through the decision-making rules and fuzzy sets established during the model creation stage. This stage allows the model to “make decisions” or recommendations based on live data.

The application stage is generally less complex than model creation because it largely involves mathematical computations rather than analytic design. These computations are typically handled by computers, enabling rapid processing of large amounts of data. For example, a traffic control model might continuously receive data about vehicle flow, using this data to adjust traffic signals in real-time.

During this stage, the model’s effectiveness is assessed by comparing the output decisions to expected outcomes. If the results are consistent and reliable, the model is validated; if not, further refinements may be necessary. This iterative process ensures the model’s decisions are accurate and relevant to real-world conditions.

5.6.11

What tasks are primarily handled in the application stage of a decision-making model?

- Processing input data
- Generating output decisions
- Defining fuzzy sets
- Consulting experts for rule creation

5.6.12

Model recovery is an essential part of decision-making system maintenance. In this stage, adjustments are made to the model to ensure that it remains accurate and relevant over time. As new data becomes available or as the decision environment changes, it may be necessary to update fuzzy sets, revise decision-making rules, or incorporate new input parameters. Model recovery is a continuous process that allows the model to adapt to evolving conditions.

This phase relies on feedback from the application stage to determine areas where the model can improve. For example, if an environmental monitoring system frequently misclassifies high-risk conditions, recovery efforts may involve refining the definitions within the fuzzy sets. By making these adjustments, the model can provide more reliable outputs and adapt to changing real-world contexts.

Through model recovery, decision-making systems remain effective and valuable tools for organizations. This iterative refinement process ensures that the model continues to meet its objectives, improving both its accuracy and flexibility over time.

5.6.13

What is the primary purpose of model recovery in decision-making systems?

- To update the model for new conditions
- To eliminate all decision-making rules
- To prevent input of new data
- To create initial fuzzy sets

5.6.14

Many applications of the presented algorithm have been published. For example:

- Klimes, C. Model of adaptation under indeterminacy. *Kybernetika*, Volume 47 (2011), Number 3, Pages 356 – 369. Click here for full text.
- Klimes, C. & Bartos, J. IT/IS security management with uncertain information. *Kybernetika*, Volume 51 (2015), Number 3, Pages 408 – 419. Click here for full text.
- Klimeš, C., Krajčík, V. & Farana, R. Proposal of Complex Software Applications. *Software Engineering Trends and Techniques in Intelligent Systems, Advances in Intelligent Systems and Computing*, vol. 575, 2017, pp. 53 – 61. DOI: 10.1007/978-3-319-57141-6_6. Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 3. ISSN 2194-5357, ISBN 978-3-319-57140-9 (WOS 978-3-319-57141-6). Click here for full text.
- Frankeová, M., Farana, R., Formánek, I. & Walek, B. Fuzzy-Expert system for customer behavior prediction. *Advances in Intelligent Systems and Computing*. Volume 764, 2019, Pages 122-131. DOI: 10.1007/978-3-319-91189-2_13. 7th Computer Science On-line Conference, CSOC 2018; Zlin; Czech Republic; 25 April 2018 through 28 April 2018; Code 213719, ISSN: 21945357, ISBN: 978-331991188-5. Click here for full text.

Case Studies

Chapter 6

6.1 IT/IS security management with uncertain information

📖 6.1.1

1. Introduction

Risk analysis is an essential part of IT/IS security management. For this, two basic types of methods are usually distinguished: qualitative and quantitative. The reliability of qualitative methods, a typical and probably the best-known representative of which is the method of targeted interviews (Delphi method) based on a set of targeted and specific questions, is derived from the subjective assessments specified during the evaluation process.

On the other side, quantitative methods are, as a rule, based on mathematical formulae expressing risk in the form of numerical values. Thus, the results (the estimates of the risk) generated by the quantitative approaches are exact values. It is however important to realize that the problem of subjectivity is only transformed - the values of impacts associated with risks (the numbers) are highly subjective values because the respective formulae are based on empirical assumptions of experts responsible for the risk assessment. This is also why computed results are usually difficult to interpret.

The individual quantitative approaches mutually differ by the complexity of the used formulae and the number of variables appearing in them]. Nevertheless, although in different ways, all of them reflect the fact that the mechanism of a risk application proceeds in the following manner: a vulnerability makes it possible that a threat overcomes countermeasures and causes harm to an asset. In fact, the asset's price motivates threat activation and the threat elevates the vulnerability. This is why the asset must be protected against threats by implementing countermeasures.

The advantages and disadvantages of the both above-mentioned approaches (i.e., quantitative and qualitative) lead us to propose their combination that will be applied in this paper to the design of a complex system evaluating the risk management of information security. The proposed method, like most of the other combined approaches, uses semi-qualitative evaluations as input variables and calculation formulae for risk evaluation. The approach is based on a proper uncertain information processing form of implicational rules; their uncertainty is expressed by the grade of the vagueness of linguistic terms ("very high", "not so accurate", "nearly" etc.). To formalize this vagueness in lexical constructs, a fuzzy mathematical instrument is used, and the inference mechanisms are based on the principles of fuzzy logic. However, a detailed explanation of these principles is not included in this paper; for this, the reader is referred to.

In the described software implementation, fuzzy logical computations are realized with the help of the LFLC 2000 (Linguistic Fuzzy Logic Controller) tool. The main goal of this paper is to describe a model that is a specialization of a general

mode (presented in the previous chapter). It is implemented in the comprehensive application shell in the way that several IF-THEN-rule knowledge bases cooperate to minimize all the shortcomings of usual approaches to risk analysis (and therefore it can be used outside the information security area). Since the model uses the knowledge expressed in vague language, the necessary data does not need to be collected by experts - the identification of assets and their definition is done in a vague and common language that is understandable to everybody. Moreover, as it was empirically proved in applications when used by the experts, the risk analysis is conducted in a surprisingly quick, clear, easy and comprehensive way, with the ability to support the understanding of the processes and results to all of the users.

This case study is organized simply and clearly. In the next section, we describe the overall structure of the system, while Sections 3 - 6 are devoted to its individual subparts; in these sections, the implementation and realization of the built-in processes is described. In Section 7 the reader can find some experimental results.

📖 6.1.2

2. Decision-making model

To make sure that the following description of the system is well understood, let us specify the meaning of the main concepts usually used in the area of information security management:

An asset is anything that has value to the organization that can be depreciated by exposure to threats. By threats, we understand potential events, which, when they turn into reality, may cause an undesirable incident and may harm an organization or system. A vulnerability is the weakness of an asset (or group of assets) that allows it to be exploited and harmed by one or more threats. The vulnerability is an attribute of an asset and represents the sensitiveness of the asset with respect to threats.

The concept of risk is connected with an event that can occur, with the probability of its occurrence, and with its potential impact. It is always future-oriented: it relates to the impact that can appear in the future. To diminish the negative impacts, protective measures or countermeasures are usually realized. By these terms, we understand processes, procedures, technical or legal means, or anything else that is designed to mitigate the effect of threats by reducing vulnerability or the impact of threats. Therefore, the threat in our model has two attributes, that represent the level of the vulnerability of the asset when exposed to the threat and the frequency of threat occurrence.

As stated above, the described decision-making model is derived from the general model (presented in the previous chapter). In fact, it is its specialization for application in the area of information security management. The decision-making process is specialized and decomposed into the following six subprocesses (see also Figure 1).

1. Identication of assets with the help of an input questionnaire.
2. Identication of the relevant threats together with their attributes. This is done by a simple expert system based on the asset type and their prices (for more details see below).
3. The risk evaluation is done by a fuzzy expert system that realizes the main part of the decision-making process.
4. The design of suitable countermeasures to protect against the identified threats is done by another simple expert system.
5. The most suitable countermeasures to reducing risk are selected by another simple expert system. The selection is done concerning to their efficiency and implementation costs.
6. Risk evaluation and the suitable countermeasures are visualized in the customized component model of the system assessed.

The whole model works in the following way: in the first step, the user identifies assets and their attributes (especially price and type, which are described using vague terms from a common language). This means that in the first step the complete description of all the assets (asset price, asset type, and dependency { the weight laid on an asset) is set up.

In the second step, relevant threats are assigned to each asset by the first expert system (the set of all threats is stored in its knowledge base).

Subsequently (step 3), using the given inference rules, the risk levels of the individual threats are evaluated. This means that their characteristics "frequency" and "vulnerability" are assigned with the help of the knowledge base. For this, it is necessary to evaluate the respective risk for all the individual threats of all of the individual assets.

At this point, it may be of interest to note that this will later enable us to select suitable countermeasures restraining the identified threats.

The next subprocess (step 4) conducts the assignment of appropriate countermeasures based on the risk evaluation, asset type, and individual threat (as every threat linked in the preceding step to any asset has its own risk level).

The following subprocess (step 5) is the selection of the most suitable countermeasures that reduce the risk to an acceptable level, with respect to their efficiency and implementation price.

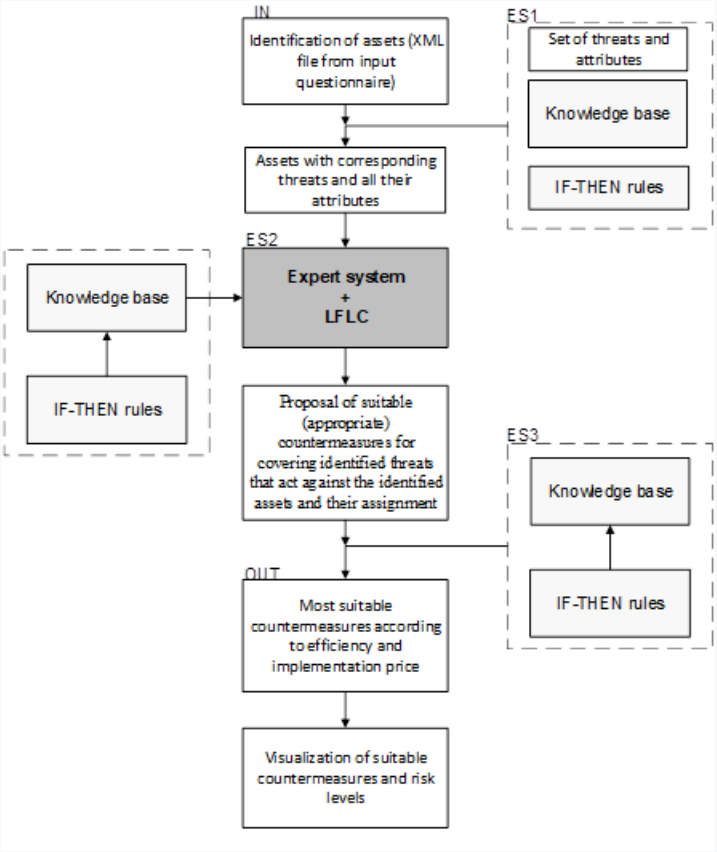


Fig. 1. Structure of the system.

The final subprocess (step 6) presents the risk evaluations and suitable countermeasures in the form of the customized component model, where assets are represented by components and countermeasures are represented by subcomponents.

🕮 6.1.3

3. Identification of assets

As said above, the purpose of the first step of the whole process is to identify all the important assets with the help of an input questionnaire. Through this questionnaire, users de

ne the assets that occur in their organization and all the attributes of such assets. An example of such an input questionnaire is depicted in the following Figure 2.

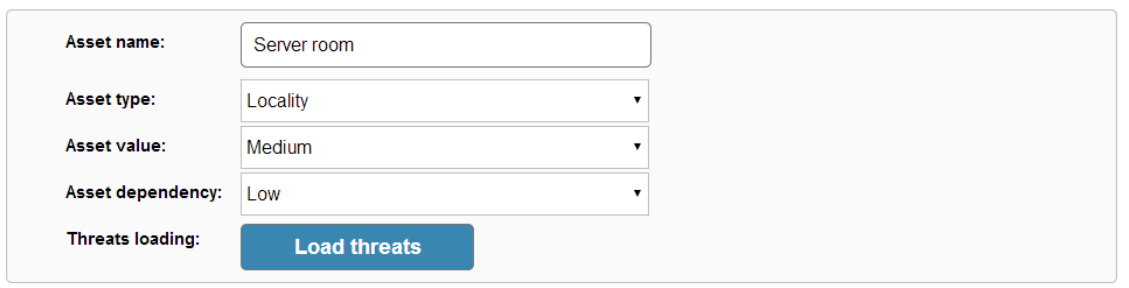


Fig. 2. Asset definition.

The figure shows one identified asset called "Server room". The asset is of "locality" type { in the questionnaire the selection of the asset type is implemented in the way that the user selects an element from the predened list of asset types (i. e. LAN, Server, Data asset,. . .). The price of the "Server room" is a fuzzy linguistic value "medium" and represents the vague evaluation of the asset price. Further, the questionnaire contains another fuzzy linguistic variable called dependency. The dependency on the "Server room" asset is identified as "low". The dependency represents the weight laid on a given asset - this enables us to specify the importance of the asset for an organization, to specify whether it is critical or unimportant. There is a button "Load threats" in the input questionnaire that initiates the expert system operation. Thus, the example in Figure 2 reads that the input is the asset server room, which is a locality with medium price and low importance.

We can thus see that the questionnaire includes vague information expressed in a common language, which is dened by linguistic values. Input fuzzy variables are modeled in the LFLC 2000 tool, an example of which, together with the form of fuzzy sets, can be graphically seen in Figure 3.

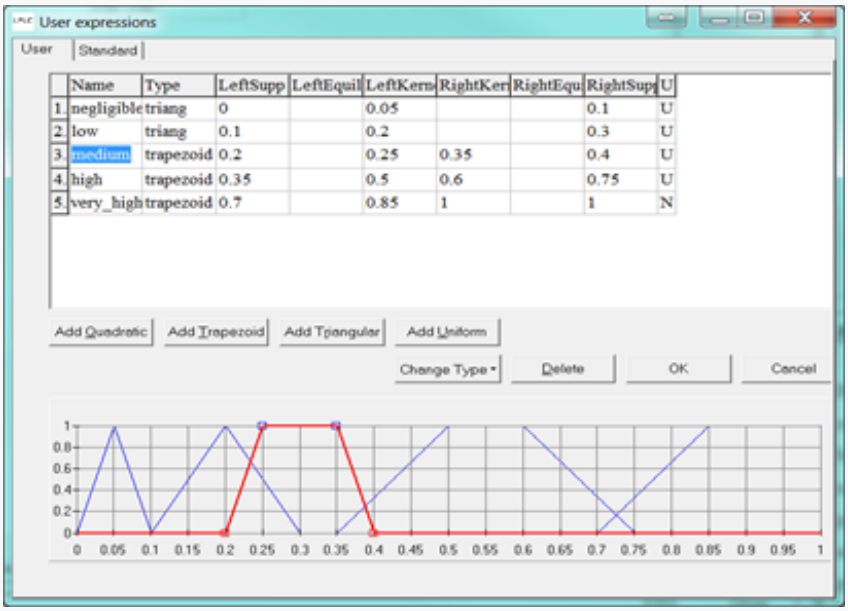


Fig. 3. Fuzzy variable "asset price".

As depicted in Figure 3, the fuzzy sets representing linguistic values may have both triangular and trapezoid forms. These forms of sets, i.e., their kernels (the set where the membership function equals 1) and supports (the set where the membership function is bigger than 0 and lower than 1) were dened by experts in the problem domain and fine-tuned during testing.

🕮 6.1.4

4. Assigning relevant threads

Having obtained records of assets as a result of the activity described in the preceding section, we will in this section now describe how threats are assigned to the individual assets based on their type. The asset records are extended with two characteristics (frequency and vulnerability) - every asset can have, and usually does have, multiple corresponding threats where every occurrence of any threat comes with one frequency attribute and one vulnerability attribute):

- Vulnerability - the fuzzy linguistic variable that defines the level of asset vulnerability in the case of the threat occurrence (values: "very low", "low", "medium", "high", "very high").
- Threat frequency - the fuzzy linguistic variable (values: "occasional", "low", "medium", "high", "extraordinary").

This extension of asset records is done by the first simple expert system (see block ES1 in Figure 1) consisting of 108 IF-THEN rules. The form of such rules is: IF an asset type THEN Threat(frequency of the threat/vulnerability level).

The threats are brought from the ISO/IEC norms, which ensure for instance the usability of ISO certifications and implementations, and compliance with Czech cybernetic laws. Let us present a couple of examples of IF-THEN rules for the selection of relevant threats.

```
IF asset type="User access to the IS" THEN {threat="Operation
error"(frequency="high", vulnerability="very low")}
IF asset type="Software" THEN {threat="Unauthorised use of
equipment by foreign entities"(frequency="extraordinary",
vulnerability="very low")}
IF asset type="Data storage" THEN {threat="User
error"(frequency="low", vulnerability="low")}
```

For instance, the first rule means, that when the asset of "User access to the IS" type is identified, the threat "Operation error" with the frequency attribute "high" and the vulnerability attribute "very low" is selected by the expert system.

📖 6.1.5

5. Computation of risk values

In this step, risks are evaluated with the help of the second set of fuzzy IF-THEN rules. This is realized in the main part of the proposed system formed by the fuzzy expert system (see block ES2 in Figure 1) whose knowledge base consists of 625 IF-THEN rules, examples of which follow:

```
IF asset price="low" AND threat (frequency="low",
vulnerability="low") AND dependency="low" THEN
risk="acceptable"
IF asset price="low" AND threat (frequency="low",
vulnerability="medium") AND dependency="low" THEN
risk="acceptable"
IF asset price="low" AND threat (frequency="low",
vulnerability="high") AND dependency="low" THEN
risk="acceptable"
IF asset price="medium" AND threat (frequency="medium",
vulnerability="low") AND dependency="low" THEN
risk="acceptable"
```

```
IF asset price="medium" AND threat (frequency="medium",
vulnerability="medium") AND dependency="medium" THEN
risk="very low"
```

All of these rules were formulated by problem-domain experts and were tested on several system prototypes and risk analysis projects. The output of this expert system, i. e. the evaluation of the considered risk, is a value of the linguistic variable "risk" (i.e. "acceptable", "low", "medium", "high"). The rules have the following form: IF asset price AND threat (frequency, vulnerability) AND dependency THEN risk. This expert system is realized in the LFLC tool.

The fuzzy expert system employed in this step uses CNF (Conjunctive Normal Form) fuzzy approximation as the inference method and the COG (Simple Center of Gravity) as the defuzzycation method (these were selected after extensive testing). CNF approximation and COG defuzzycation are applied by the LFLC for the risk evaluation. The LFLC tries to nd IF-THEN rules, that can be applied. Based on the deuzication method, the LFLC itself selects the most appropriate IF-THEN rule and applies it. The output fuzzy value is therefore either an initial (basic) value (acceptable, low, medium, high) or a composition of some basic value extended by a fuzzy operator (very low, very high).

🕮 6.1.6

6. Selection of the most suitable countermeasures

The proposal of both possible and most suitable countermeasures is conducted by another expert system (see block ES3 in Figure 1). All identified threats that act against some of the assets must be eliminated. The countermeasures are suggested by another set of fuzzy IF-THEN rules. So, analyzing the vague input information (acquired from the input questionnaire), this expert system assigns relevant countermeasures to individual assets (together with dening their price and effectiveness).

First, the relevant countermeasures are determined. The relevance is detected with the help of two knowledge bases { two sets of IF-THEN rules { one describing the affinity

of countermeasures to asset types and second describing the affinity of countermeasures to threats.

As an example, let us present a snippet of one rule from the first knowledge base applicable to the identified asset "switch" (defined as an asset of "LAN" type):

```
IF asset type="LAN" THEN select COUNTERMEASURES {"Equipment
siting and protection"; "Supporting utilities"; "Cabling
security"; "Equipment maintenance"; ...; "Documented operating
procedures"; "Change management"; "Fault logging"; ...;
"Equipment identification in networks"; ...; "Key management";
"Control of technical vulnerabilities"; }
```

The second set of rules may be illustrated by the following snippet (for threat "Remote spying" that was selected for asset "switch"):

```
IF threat="Remote spying" THEN select COUNTERMEASURES
{"Cabling security"; "Network controls"; ...; "Policy on use
of network services"; "User authentication for external
connections"; "Equipment identification in networks"; "Key
management"; "Control of technical vulnerabilities";
"Reporting information security events"; ... }
```

The relevant countermeasures are those that are generated by both considered knowledge bases. The reason for considering only the intersection of results generated by the two expert systems is to provide the relevant and suitable countermeasures as a response to the individual threat and individual asset type (one threat can manifest against multiple assets, that don't share the same characteristics - by the application of the intersection, the system will not present countermeasures that are not relevant for the identified individual asset).

Following this, the most appropriate countermeasures are chosen. The knowledge base conducting this selection is again constructed in the LFLC tool. Examples of IF-THEN rules for countermeasures selection (where the attributes price and efficiency are specified in the braces):

```
IF asset price="very low" AND risk="low" THEN intersecting
countermeasures WITH (price="low", efficiency="low")
IF asset price="medium" AND risk="low" THEN intersecting
countermeasures WITH (price="low", efficiency="medium")
IF asset price="medium" AND risk="medium" THEN intersecting
countermeasures WITH ((price="low", efficiency="high")
```

The outputs and results of the risk analysis conducted by our set of expert systems is a visualization (step 6) that uses a modified component model, where the components represent assets and the subcomponents represent the countermeasures relevant to the assets. The colors of the components represent the

risk value of the asset and the colors of the subcomponents represent the effectiveness of the selected and proposed countermeasures.

🕮 6.1.7

7. Comparison and results

Let us demonstrate the whole process on a simple example. Naturally, because of the extent of the risk analysis procedure we will use only a snippet of it. The usual approach is that an expert proposes the evaluation criteria (numeric evaluations of assets, threats and their attributes). Then the expert collects information from users, usually in the form of a table - such table contains the necessary attributes that are being used in the formulae for the risk calculation.

Using a classical approach, the expert identifies an asset and specifies risk which is computed using a simple equation:

A(x) * T(x) * V (x);

where

- A is the asset price (from a range of 1-4),
- T is the probability of the threat (from a range of 1-5),
- V is the value for vulnerability (from a range of 1-5),
- x is the particular asset.

As an example consider an asset "switch". The expert tries to identify a set of threats (depending on the time the expert has, his/her expertise and the strategy he/she chooses the set may wary). Respondents of targeted interviews evaluate the price of asset "switch" (let it be in our example the number 2) and in cooperation with the expert they evaluate the values for vulnerability and probability of each identified threat. Let us assume that they identified three threats and their attributes are evaluated as follows:

- infiltration with vulnerability value 2 and probability value 2,
- hardware malfunction with vulnerability value 1 and probability value 1, and
- maintenance error with vulnerability value 1 and probability value 1.

For every threat, the risk value is computed according to the above presented formula. In the considered example the expert gets values { infiltration = 2 * 2 * 2 = 8; hardware malfunction = 2*1*1 = 2; maintenance error = 2*1*1 = 2. Now the expert determines a threshold value (let it be in our example 5) determining that every risk

value above this threshold must be covered by countermeasures. After this evaluation, the expert suggests the countermeasures according to his domain knowledge.

To get the corresponding results, our approach uses the terms from a vague common language, so there is no explicit scale of numeric values or need for its approval. Also the collecting of the data for analysis is not provided by experts, but by users using an input questionnaire.

In the first step, the users identify the asset "switch" (asset type "LAN") and assigns to it a price ("medium") and dependency (it is a core switch, so the dependency is "very high").

Following this (in step 2), the system provides all the relevant threats and their corresponding attributes (selecting only the relevant one from the whole knowledge base), the snippet of the relevant threats may be:

```
"Disturbance" (frequency = "very low"; vulnerability =
"medium")
"Breach of maintainability" (frequency = "low"; vulnerability
= "low")
"Remore spying" (frequency = "medium"; vulnerability =
"medium")
"Theft" (frequency = "occasional"; vulnerability = "medium")
"Dust, corrosion, freezing" (frequency = "occasional";
vulnerability = "very low")
```

In the third step, the system evaluates risk levels of all the provided threats. It may happen that some of the treats may appear in connection with several assets and with different attributes. This is because, the threats can represent different events (e.g. threat "Theft" can represent theft conducted by the internal sta or theft conducted by an external thief). So, the risk levels of threats corresponding to the identified asset "switch" may be:

```
"Disturbance" - risk="medium";
"Breach of maintainability" - risk="low";
"Remore spying" - risk="low";
"Theft" - risk="medium";
"Dust, corrosion, freezing" - risk="low";
"Destruction of wiring" - risk="low".
```

In the considered example of a "switch" asset and the above evaluation, the system proposes countermeasures for the threats and the asset (according to asset type) by the intersection of sets of countermeasures linked to the asset (see Section 6).

The fifth step of the system selects only the most suitable countermeasures. Because not a single risk is evaluated "acceptable", the system is processing all the threats from the previous step. Because of the asset price being "medium" and not a single risk being more than "medium", the system excludes all the countermeasures with price "high" and all countermeasures with "low" efficiency regardless of their price. The snippet of the final list of recommended countermeasures may be like this:

```
"Equipment siting and protection" (price="small",
efficiency="big")
"Supporting utilities" (price="small", efficiency="big")
"Cabling security" (price="small", efficiency="big")
"Equipment maintenance" (price="small", efficiency="big")
"Network controls" (price="small", efficiency="big")
"Security of network services" (price="small",
efficiency="medium")
"Monitoring system use" (price="small", efficiency="big")
"Segregation in networks" (price="small", efficiency="big")
"Network connection control" (price="small", efficiency="big")
"Network routing control" (price="small", efficiency="big")
"Key management" (price="small", efficiency="big")
"Control of technical vulnerabilities" (price="small",
efficiency="big")
```

📖 6.1.8

8. Conclusion

The presented case study deals with the application of a model for decision-making under uncertainty for implementation of complex software tools, such as information security risk management - the tool is published at the web URL http://ar.proit.cz. The results have shown that the model proposed in the previous chapter is suitable for these tasks. The experience gained is reflected retroactively to the expert systems rules. An expert on information security risk analysis is no longer necessary to perform the analysis. The results are implemented in fully matured applications, that are ready for usage in the problem domain. The decision-making model itself is very versatile and can be used in other domains - today, the problem domain of searching over dynamic and static content and the problem domain of operation system designs are being implemented.

Source: Klimes, C. & Bartos, J. IT/IS security management with uncertain information. *Kybernetika*, Volume 51 (2015), Number 3, Pages 408 – 419. Click here for full text.

6.1.9

Which thread is correct for this case study?

- IF asset type="Data storage" THEN {threat="User error"(frequency="low", vulnerability="low")}
- IF asset type="Data storage" THEN {threat="System error"(frequency="high", vulnerability="high")}
- IF asset type="Data storage" THEN {threat="Memory error"(frequency="middle", vulnerability="high")}

6.2 Proposal of complex software applications

6.2.1

1. Introduction

When designing complex software applications, there are frequently issues that can be described by the following characteristics:

- the problem is not algorithmizable,
- the problem is new, not repeatable, unique, usually very important,
- there are more factors influencing the solution (cannot be expressed by numbers),
- some of the factors are not known at all or there are complicated relations,
- changes of some elements in the software application environment where the solution is carried out are random,
- there is no routine solution,
- there are no analytical methods to find an optimal solution,
- there are more criteria to assess the solution, some of which are of a quantitative nature,
- interpretation of information necessary for the decision is difficult,
- a human is usually an active element of the system (they create and change the system by deliberate activity).

Thus, we have to decrease the risk in a proposal of a solution of a **badly structured problem**, when a wrong decision of the project team can result in creating an unsuitable application or even in an application which does not meet customer's

idea. It is then necessary to decrease the risk of a wrong decision to the lowest possible level. It primarily concerns SW applications are crucial from the point of view of maintaining company strategy in the future, i.e. so-called **strategic applications**.

The solution is a proposal of an automated tool which enables to substitute an expert when selecting a suitable action based on symptoms of the current way of work. This tool will suggest which practices are suitable to use in SW development. Therefore, the tool supports decision making on practices relevancy, whereas implementation itself is up to the team. This step only reduces the risk of selecting an unsuitable practice, but not the risk of unsuitable implementation and misunderstanding the principle in the practice background. The tool cannot eliminate misunderstanding so support of an experienced member of the team or a mentor is necessary.

In order to propose a suitable decision-making tool, we have to define the decision-making process, input and output data, data base and rules. Simulation of decision-making processes are specific in the following:

- The decision making does not rely only on analytical information, but mostly on knowledge represented by a recognition process and a process of abstraction (which is a privilege of brain activity).
- The decision can be made by several approaches considering how many individuals will assess it.
- It is very difficult to accurately define the algorithm of making a decision.
- Great part of information used in decision making is of external origin with respect to already established and known data base of the decision-making problem.

A decision-making process, in a wider sense, can be defined as an organic unit of three phases:

- Information (knowledge acquisition).
- Planning (considering the alternatives).
- Selecting (selection of a variant).

Making a decision relates only to the last phase, the first two phases (knowledge acquisition and considering the alternatives) are only preparative ones.

In order to define the structure of a decision-making process and thus to create prerequisites for finding effective processes for its algorithmization, we have to deal with decision-making processes in a wider, primarily methodological, perspective. One of characteristic features of decision-making processes is the fact that they often work with indeterminate (but quantitative) information, which often results from the fact that the input quantities of these processes are defined by a human based on their estimates, experience, opinion, etc.

Designing complex software tools can be advantageously supported by using modelling tools based on fuzzy algorithms and principles used in modelling of decision-making processes. Nevertheless, it is necessary to dispose of a system enabling to acquire knowledge from reactions of users of these complex systems. It primarily concerns simulation of decision-making processes as well as ways of knowledge acquisition.

Implementation of such a decision-making system was presented in the previous chapter. Following this general decision-making model, the case study discusses its use in the process of proposing complex software tools. An example is provided in a form of a proposal of a safe operating system architecture.

🕮 6.2.2

2. General decision-making model

This decision-making model stems from the structure presented in previous chapter. Elements of the decision-making process were divided in the following groups:

- S – set of situations,
- D – set of all possible solutions,
- G – set of all objectives (admissible) of subsequent functioning of the given system,
- F – set of all existence levels (probabilities) of the given object,
- K – set of all assessments of the given solution,
- T – time interval.

The decision-making process itself is represented in various mappings between those sets. It primarily concerns the following mappings:

1. process of information completion about the given situation and its assessment, i.e. selection of only such information that is relevant for the final solution:

$$M_1 : S \times T \times F \rightarrow S \times T \times F \qquad (1)$$

2. process of creation of all admissible solutions which consists of two partial processes

$$M_2 = M_{22} \circ M_{21} \qquad (2)$$

where

- M_{21} – formulating objectives of the control based on the description of the given situation $M_{21} - S \times T \times F \rightarrow G \times S \times T \times F$
- M_{22} – formulating admissible solutions $M_{22} - G \times S \times T \times F \rightarrow D \times S \times T \times F$

3. process of modelling of all effects of admissible solutions:

$$M_3 : D \times S \times T \times F \rightarrow D \times S \times T \times (S \times T)^* \times F \quad (3)$$

where $(S \times T)^*$ determines the set of all strings over $(S \times T)$, (each admissible solution is assigned with a set of situations and their time courses which are created based on the given decision).

4. process of approving the solution itself which consists of two partial processes

$$M_4 = M_{42} \circ M_{41} \quad (4)$$

where

- M_{41} – assessing the behavior of the effects of admissible solutions $M_{41} - D \times S \times T \times (S \times T)^* \times F \rightarrow D \times K \times T \times F$,
- M_{42} – selection of the best variants $M_{42} - D \times K \times T \rightarrow D \times T$.

The whole decision-making process consists of a gradual composition of the following processes,

$$M = M_4 \circ M_3 \circ M_2 \circ M_1 \quad (5)$$

as can be depicted in the following diagram (see Fig. 1).

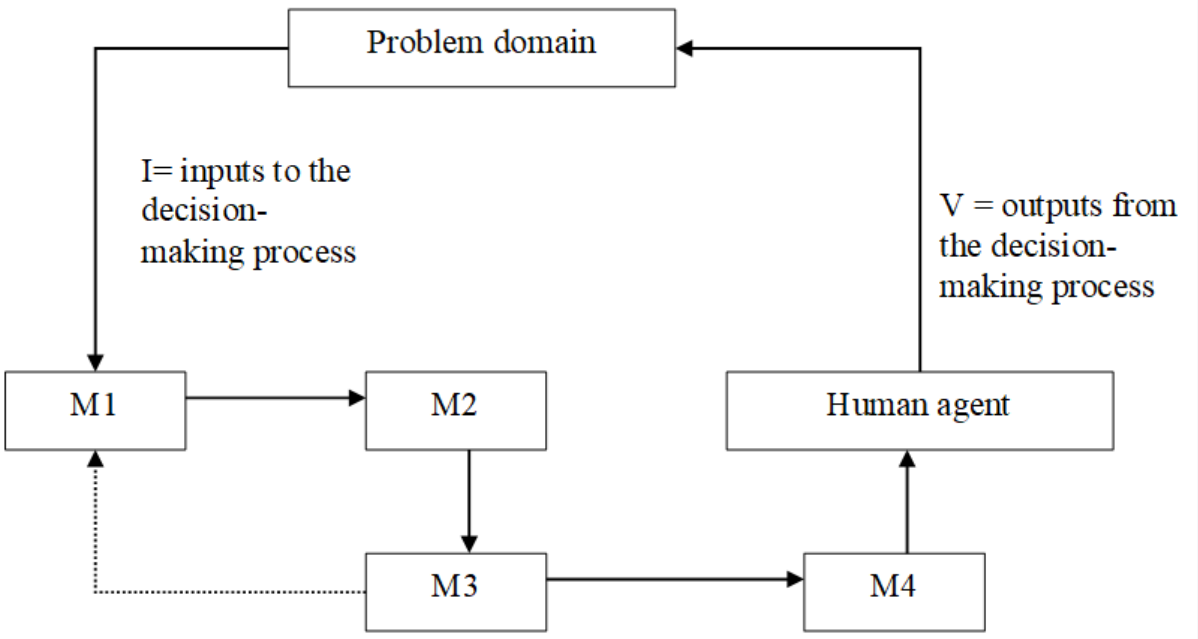


Fig. 1. Decision-making process

Note that implementation of processes M_1 – M_4 in the general system was carried out by using so-called fuzzy algorithms with the results of the fuzzy sets theory.

🕮 6.2.3

3. Model of creating a complex software application

When creating a software application, it is very important to encompass and analyze the key requirements and needs of the future software. Those requirements can be stored in various forms (document with a text description of the requirements, visions containing identification of the key system functionalities depicted by use-case diagrams, etc.). Our objective is to process vague (indeterminate) information created during the analysis of the key requirements and functionalities of the future software.

The above-presented decision-making model is fully usable for creation of complex software applications. Assume that the proposed software application is an information system providing information from a local database in a form described by a user using rules. Those can be described vaguely, most often by linguistic terms characterizing the size of specific quantities or relations between specific quantities. In order to implement those inputs and algorithms on a computer, it is necessary to use a suitable mathematical apparatus. One of the possibilities is the use of the fuzzy sets theory.

Let's consider an example of input information in a decision-making process about selecting the most important processed information by a software application, which can be divided into the following groups:

- amount of data;
- content of information;
- importance;
- availability;
- credibility;
- information costs and acquisition
- time necessary to acquire the information;
- size from the point of view of space taken up on disks, etc.

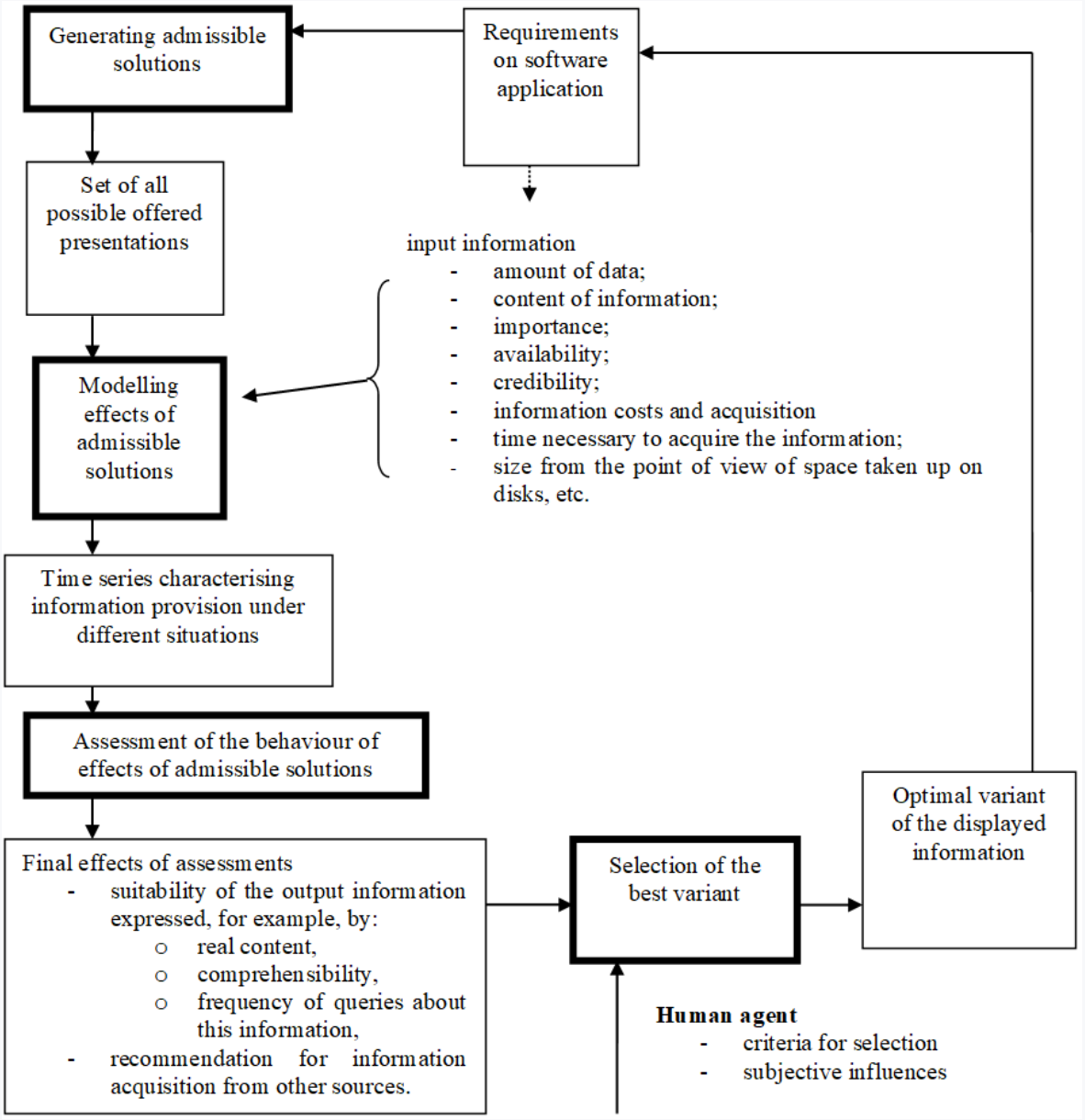


Fig. 2. Structure of the decision-making system

The output of the proposal system should be a set of all possible solutions represented by the following characteristics:

- suitability of the output information expressed, for example, by:
- real content,
- comprehensibility,
- frequency of queries about this information,
- recommendation for information acquisition from other sources.

The whole decision-making process is shown in the Fig. 2.

🕮 6.2.4

4. Decision-making model for a proposal of a safe operating system architecture

A model for a proposal of architecture of a safe operating system stems from the general model for decision-making support under indeterminacy presented in chapter 2.

Process M_1 represents completion, i.e. input data completion which relate to architecture of an operating system, and selection of relevant components of an operating system which are important for the final solution.

Input data completion is realized on the basis of a questionnaire, which is answered by the user (operator) of the expert system. A general architecture of an operating system (a set of all components) is then specified and only components relevant for the final solution are selected.

Process M_1 is expressed as:

$$M_1: K \times A \rightarrow K^* \qquad (6)$$

where:

- K – a set of all components of an operating system which enter the decision-making process. It represents a general architecture of an operating system.
- A – a set of answers which describe actual architecture of an operating system. It enters the decision-making process through a questionnaire.
- K^* – the final set of components after information completion. It represents the real architecture of an operating system.

Process M_2 generates all admissible solutions. It has 3 sub-processes which create the following (based on security requirements):

1. a set of operating system components that are subject to protection,
2. a set of threats that the components are expose to,
3. a set of vulnerabilities relating to the components and threats.

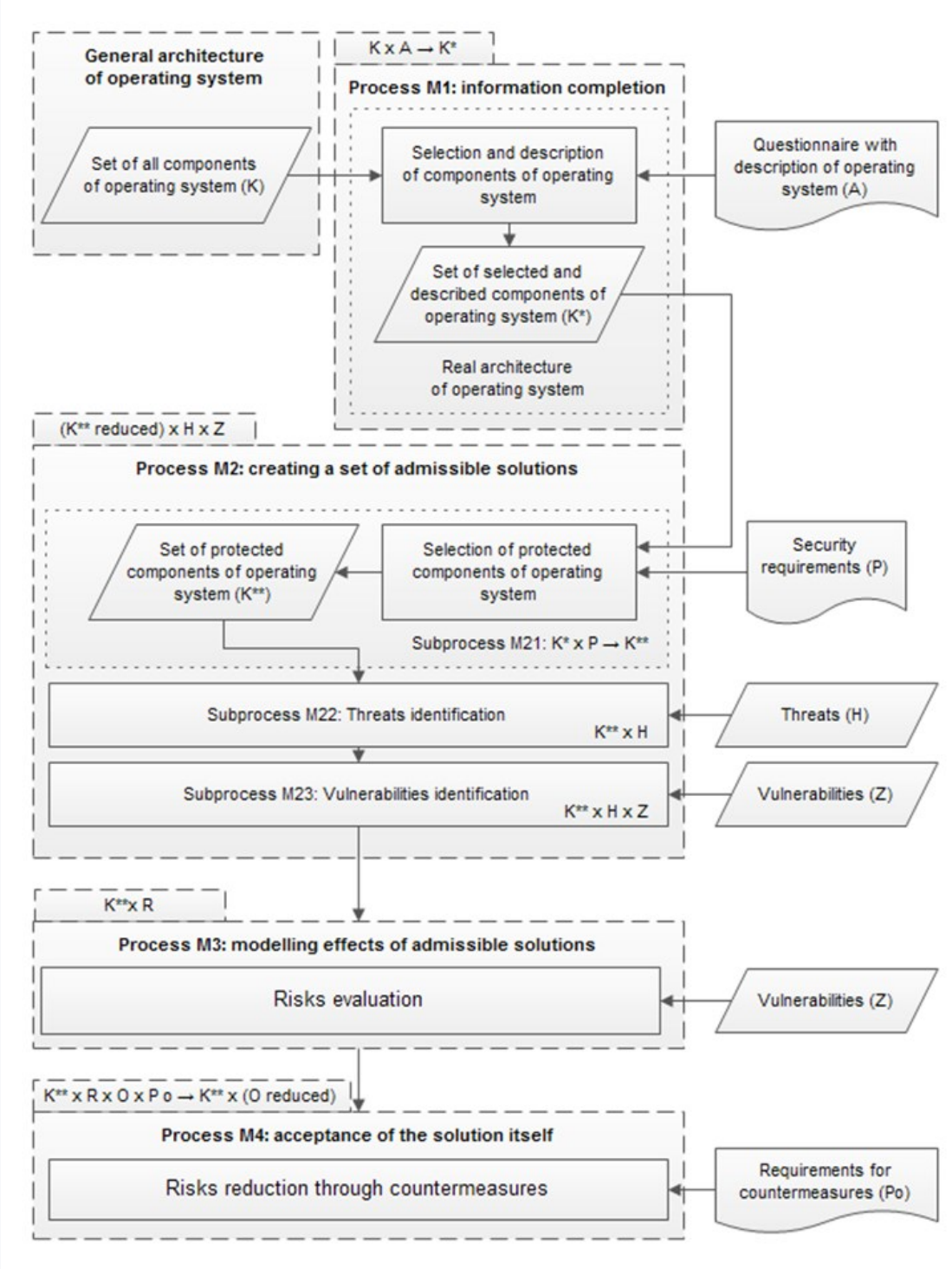


Fig. 3. Model of a proposal of a safe operating system

Process M_{21}

Based on security requirements, this process consists in creating a set of operating system components that are subject to protection. Identification of these components is realized using rules from a knowledge base. Process M_{21} is expressed as:

$$M_{21}: K^{*} \times P \rightarrow K^{**} \quad (7)$$

- where:
- P – a set of security requirements laid on the operating system by the user. They enter the decision-making process using a questionnaire.
- K^{**} – the final set of operating system components which are subject to protection (protected operating system components respectively).

Process M_{22} represents creating a set of threats affecting the operating system components which are subject to protection. Their identification is realized on the basis of a knowledge base depending on security requirements. Process M_{22} is expressed as:

$$M_{22}: K^{**} \times H \quad (8)$$

where: H – the final set of threats which the components are exposed to, affected by respectively.

Process M_{23} consists in creating a set of vulnerabilities relating to the protected operating system components and their threats. Vulnerability identification is realized on the basis of a knowledge base depending on threats, absence or insufficiencies of appropriate security measures (they are part of an operating system too).

It holds that a threat takes advantage of vulnerability and if there is no threat to take advantage of such vulnerability, there is no risk of breaking security of an operating system. Thus the set of operating system components subject to protection shrinks. Process M_{23} is expressed as:

$$M_{23}: (K^{**}narrowed) \times H \times Z \quad (9)$$

where: Z – a set of vulnerabilities relating to the components and their threats.

Process M_3 models effects of admissible solutions. Based on the rules from the knowledge base, it qualitatively assesses risks for the components subject to protection. A security risk is assessed for individual threats to each protected component. The level of such a risk determined on the basis of probable occurrence of the threat and the extent of its impact on the protected component, where:

- the probability of threat occurrence can be determined in relation to the number of vulnerabilities or attributes of the component. It rises if user's IT knowledge is lower, if there are several users, or if the operating system works in unsecured computer network without appropriate security measures.
- the threat impact on the component is assessed by expert.

Process M_3 is expressed as:

$$M_3: K^{**} \times R \quad (10)$$

where: R – a set of risks affecting these components.

Process M_4 selects the most suitable measures for operating system components subject to protection. The purpose of such protective measures is to reduce or eliminate security risks existing for the protected components. Identification of protective measures is realized on the basis of rules from a knowledge base depending on threats which the components are exposed to.

Selection of all existing protective measures for particular components narrows on the basis of user's requirements (e.g. level of configuration and use of an antivirus system: easy – advanced – expert, etc.)

Process M_4 is expressed as:

$$M_4: K^{**} \times R \times O \times P_O \rightarrow K^{**} \times (O\ narrowed) \quad (11)$$

where:

- R – a set of risks affecting these components.
- O – a set of all protective measures relating to the components and their threats.
- P_o – a set of user's requirements on the protective measures. They enter the decision-making process through a questionnaire. They are a base for further narrowing the set of protective measures for individual components.

📖 6.2.5

5. Conclusion

The presented case study solves implementability of a model of decision making under indeterminacy for implementation of software tools, such as a proposal of a safe operating system architecture. The results showed that the general model proposed in the previous chapter is suitable for these tasks.

Source: Klimeš, C., Krajčík, V. & Farana, R. Proposal of Complex Software Applications. *Software Engineering Trends and Techniques in Intelligent Systems, Advances in Intelligent Systems and Computing*, vol. 575, 2017, pp. 53 – 61. DOI: 10.1007/978-3-319-57141-6_6. Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 3. ISSN 2194-5357, ISBN 978-3-319-57140-9 (WOS 978-3-319-57141-6). Click here for full text.

📝 6.2.6

The whole decision-making process consists of a gradual composition of the following processes is:

- M = M4 . M3 . M2 . M1
- M4 = M4 + M3 + M2 + M1
- M = (M1 + M2) . (M3 + M4)

6.3 Fuzzy-expert system for customer behavior prediction

📖 6.3.1

1. Introduction to the problem

The core task of the expert system is to hand on the knowledge or experience of an expert to another person who does not have such expert knowledge, but needs him or her to fulfil a task in good quality. In the ordinary life, a less experienced person consults a problem with a more experienced colleague. The expert system can mediate such advice without the expert being available. Expert experience is appropriately formalized and maintained in order to be used when querying the system by the user. Knowledge preservation is useful not only for decision support, but also for planning, diagnosing or predicting future developments. The first expert systems emerged in the early 1970s.

Knowledge or experience can best be passed through a natural language, as it has the best expressivity. However, natural language is problematic for computer processing. It contains a lot of constraints arising from context or vague terms. The practice has been proven that such binding can best be addressed using IF-THEN rules together with fuzzy sets and fuzzy logic. Fundamentals of the theory of fuzzy sets were laid by Lotfi Asker Zadeh in the 1960s. The basic idea of fuzzy theory is the "degree of belonging" of an element to a set of elements. Thus, in a simplified way, apart from deciding whether an element belongs to or does not belong to a set, it also introduces a partial affiliation of the element into a set.

In this paper, we focus on using a fuzzy-expert system to predict customer behavior in retail. This model is used to determine the amount of inventory of individual products. Contribution follows the results of the research work of the research workplace.

One of the main business goals in general is to maximize product sales. It is obviously the core task of each organization that is based on prosperity, competitiveness, profitability and other attributes that determine the overall success of the organization. In order to achieve this goal, it must be ensured that the product sold is in sufficient quantity (and quality of course but it is no longer related to this work). Thus, a simple solution is available – always to have more products than can be sold in a given time period. But that is not a good solution. Storage of such goods is demanding for premises where proper conditions are necessary and costly. In inventory, the company can have also considerable amount of financial funds. Therefore, one of the goals of supply chain optimization is to minimize the inventory. Also, goods with short warranty period, typically food, cannot be ordered in this way because the customer always prefers fresh goods to the old ones. There is a risk of expiration of the warranty period and this can lead to loss of profit. The ideal thing is to have the goods as much as we are able to sell them at a certain time.

An experienced salesperson can handle this problem easily. The longer he or she works in a particular location, the more accurate he can predict and manage the amount of orders. He does it mostly intuitively. However, something can happen that the conditions are changed due to the price change of goods, traffic restrictions, bad weather conditions or change of the location, a new colleague from another department etc. In this case the old experience is not valid any more.

📖 6.3.2

2. Current inventory planning methods

There are several methods that support inventory strategic management. The two the most important ones are the ABC method and the JIT method. Both of them are based on statistical analysis over the reference period (usually over a year).

The ABC method is based on the consideration that there is no need to devote the same attention to each type of inventory. The method therefore divides inventories into three groups and work with each group in a different way.

Inventories are broken down by their share of total annual turnover. The method is also sometimes modified by the so-called XYZ analysis, which evaluates inventories in terms of smoothness and predictability of their consumption or sales. Group X means inventory with a regular and predictable consumption or sale. Group Y represents inventory with uneven but still predictable consumption or sales (e.g. seasonal fluctuations). Group Z means inventory with uneven and unpredictable consumption or sales. The inventory can then be determined as follows in Table 1.

Table 1. Product categories by ABC – XYZ method.

	X	Y	Z
A	low	low	middle
B	low	middle	high
C	middle	middle	high

The Just In Time (JIT) method belongs to the LEAN group of methods and its aim is to deliver the goods exactly at the agreed time to a precisely agreed place. JIT minimizes inventory, does not have fixed delivery dates or fixed quantities. It is characterized by very frequent deliveries. JIT does not have any set of rules or procedures, but rather a philosophical approach that was created by Toyota around 1926. Although JIT is most often used in production, many of its principles and ideas can be applied to business. The JIT expert system would be a suitable support tool for determining the amount of orders. In retail practice, two approaches are generally applied:

- First approach: we order routinely as many items as sold for the past period since last delivery.
- Second approach: we first define the maximum number of pieces. When ordering, we will find out how many pieces are missing to the maximum and order them.

Both approaches have their shortcomings. For example, the first approach fails to respond to the differences in consumption at different times and does not respond flexibly enough to the increase in demand. Yet it is more flexible than the second approach, which in turn is not able to respond to the drop in demand.

🕮 6.3.3

3. Analysis of system behavior, determination of main parameters, identification of outputs

For the design of the expert system (EC), based on previous research, it was determined that the quantity to be tracked would be the quantity of goods sold. The EC will analyze the results of sales for previous periods and, on the basis of these, will estimate future sales figures, assessing the possible factors that may affect the amount of sales.

Sales of particular types of goods primarily affect:

- advertising, promotion,
- price,
- significant days, holidays,
- seasonality,
- change in store availability,
- organizing important events near the shop (e.g. fairs, concerts, etc.),
- weather.

Each of these influences has its own specific features and impacts on expert system knowledge base. The influences are well known and therefore we will not deal with them in detail. Just only a note.

The most significant measured quantity in our case will be the difference between estimated sales and actual sales. It can be assumed that this difference will always be greater at the beginning of the prediction process than at the end. Only after the system has been debugged and after the necessary adjustments can be expected that the difference may be small. At the same time, it is understandable that permanent zero differences can never be achieved. The reason is that the sale of goods affects various random factors and influences. Definition of main linguistic variables:

- A: units(very small, small, medium, high, very high)
- B: price (very small, small, medium, high, very high)
- C: holiday(no, one day, two days, Easter, Christmas)
- D: event(yes, no)
- E: availability(yes, no)
- F: weather(very bad, bad, normal, nice, very nice)
- G: difference (very small, small, medium, high, very high)

Linguistic variables will be specified by the names of parameters and by the names of fuzzy sets. The names of the fuzzy sets will be in the brackets.

To determine the number of rules for a complete system description, we need to quantify the number of fuzzy sets for individual parameters.

$$N = n_A.n_B.n_C.n_D.n_E.n_F.n_G = 5.5.5.2.2.5.5 = 12\,500 \quad (1)$$

If we use shallow (subjective) reasoning [4]; the number of rules can be reduced by using a deep (objective) reasoning with one weight set. The evaluating process will therefore be carried out in two steps. In the first step, a second group of variables (i.e. C, D, E, F, G), whose significance for the overall result should be lower, and it will be evaluated. In the second step, the first group of variables (A, B, H) will be evaluated. To evaluate the first set of variables, a new linguistic variable called weight will be introduced, which will be the output from the first step of the evaluation process and the input into the second step.

Linguistic variable weight and its fuzzy set:

H: weight (very small, small, medium, high, very high)

The number of rules for a full system description will be significantly reduced:

$$N = n_A.n_B.n_H = 5.5.5 = 125,$$
$$n_H = n_C.n_D.n_E.n_F.n_G = 5.2.2.5.5 = 500. \quad (2)$$

So the knowledge base will not contain more than 625 rules. As part of system verification, it is possible to modify the structure of individual sets. Figure 1 shows the definition of fuzzy sets for the variable weight. On the vertical axis, there is a function of degree of membership, on the horizontal axis then the range of values that this linguistic variable can acquire.

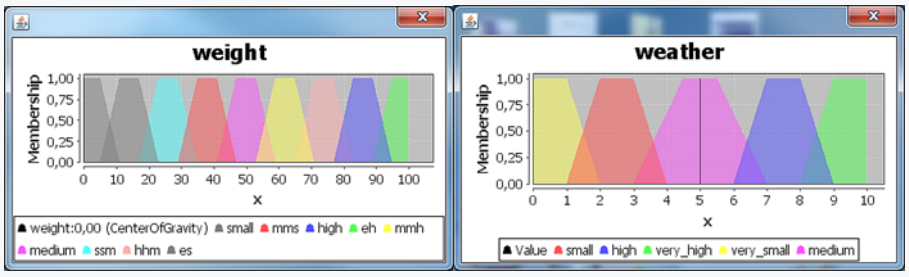


Fig. 1. Definition of fuzzy sets for variable *weight* and variable *weather*.

The data base for verification of the created prediction system was based on data from 1 June 2012 to 8 September 2014 from a specific supermarket that does not wish to be identified. This is therefore a long enough period to obtain a relevant description of the system's behavior. The monitored data are:

- the number of pieces sold,
- goods price,
- date of sale.

The following was added to the database:

- barcode,
- day of the week,
- month,
- prediction,
- the difference between the prediction and reality,
- fuzzy rule numbers enabled to analyze system behavior.

In addition, important events organized around the shop were identified where only the annual pilgrimage, organized during the second Sunday of September, was practically taken into account here. Other important events were not mentioned by shop staff.

For a fuzzy set of weather conditions, the weather was found in the ČHMÚ Homepage. The scale 0-10 was used to evaluate the weather, where 0 is very bad, 10 is very beautiful. This scale has been covered by fuzzy sets, see Fig. 1. Historical records for the area under review had to be evaluated on a daily basis by temperature, sunshine, precipitation, wind strength and season.

For the most important variable units and price, a variable context method was used, which is used at the workplace, especially in the LFLC tool. The range of actual values, minimum and maximum, and the corresponding context were always set for the tracking period.

🕮 6.3.4

4. Knowledge base, inference mechanism

The knowledge base is compiled using IF-THEN rules. Each rule consists of an antecedent and a sequencer. The antecedent forms the conjunction of the associated input language variables with their valuation, followed by the evaluation of the output variable.

The knowledge base consists of two parts, the weight parameter will be evaluated first, it will be the output of the first set of rules and the entry for the second set of rules.

Example rule for the first group:

```
IF price IS very_small AND weather IS very_bad AND holiday IS
no AND event IS yes AND availability IS yes THEN weight IS
very_small;
```

Example rule for the second group:

```
IF units IS very_small AND difference IS high AND weight IS
medium THEN tip IS very_small;
```

The rules were based on the experience of an expert working in a supermarket, whose job is ordering the goods. Based on his experience, an initial set of rules was developed. These rules were then verified and possibly modified during the test development phase of the system.

One of the most important parts of the whole system is the inferential mechanism. That is why the choice of the inference mechanism was given the greatest attention. In fuzzy systems there are two basic options. These are the logical deduction and the fuzzy approximation. The fuzzy approximation is suitable for fuzzy control because it is more about characterizing the course of function in a local area. Thus, fuzzy sets are defined to best describe the course of a function. Although this method would have been used by the solver in the area of industrial systems management and could be used for the exemplary solution, its principle is intended to achieve other values of EC behavior. Therefore, the principle of logical deduction was used for the EC. Various methods were tested for defuzzyfication in both calculation steps. Based on the test results, the Center of Gravity - COG method was chosen.

🕮 6.3.5

5. Expert system testing

To verify the suitability of possible procedures and methods, prototyping and subsequent evaluation of the results of the procedures used were chosen.

Methods of selecting data from the source database to determine the contexts of key variables have been verified. Three variants were compared:

- **m1**: set consisting of all the occurring values. Disadvantage: it was not possible to predict the maximum / minimum that has not yet occurred.

- **m2**: as in the previous point and increased by 10%. Predictions of minimum / maximum were allowed, but in the course of time only the growth of sets was made, not the estimation of scales through the reduction of sets.
- **m3**: set consisting of a selection of values and increased by 20%. The choice of values allowed for the flexibility of the sets. Because the set contained much less elements, the magnification increased to 20%.

Methods for evaluating language variables for multiple source data have also been verified:

- **o1**: valuation is the average of the values from the selection.
- **o2**: the median of choice is the rating.
- **o3**: evaluation is the last occurring value of the selection.

The graphs in Fig. 2 show a comparison of two methods. Both are applied to the same type of goods in the same period:

- **method 1**: m2, o2, trapezoid set shape.
- **method 2**: m2, o3, trapezoidal shape.

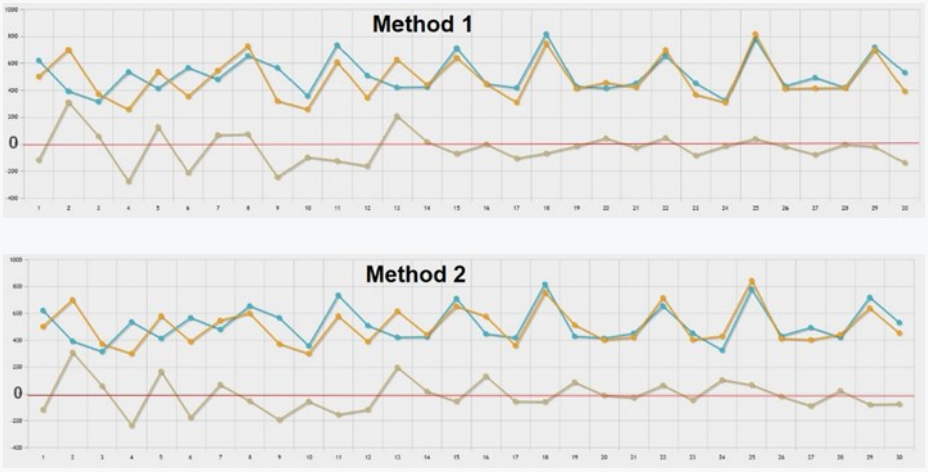


Fig. 2. Results of testing two evaluation methods for the first monitored month.

First day values were manually entered based on a qualified estimate, second and other days were already performed by an expert system. The graphs show that the estimate for the first week was evaluated in both cases in a similar way. The next weeks the estimates (tip) were refined. It can be seen on the difference value, which is the difference between the estimate and the units.

The method of calculating the standard deviation of the differences according to the following formula was used to compare the results:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (3)$$

The results obtained by all three methods are presented in Table 2. In the evaluation of other months and other methods, **method 3** with parameters obtained by means of **m3**, **o3** was chosen because it gave the best results.

Table 2. Test results of three evaluation methods for the first monitored month.

Date	Method 1	Method 2	Method 3
2. VI	308	308	308
3. VI	57	**6**	57
…			
29. VI.	36	31	**25**
30. VI	-64	**-51**	-81
sum of differences	-476	-225	**43**
number of nearest estimates	9	9	**13**
standard deviation	123,54	128,98	**118,80**
sum of absolute values	**2728**	2857	2737

The bases of the rules have been tested and edited during sales estimates over a portion of the test data base. The extreme differences between estimate and reality occurred in some days. Such cases were searched, then the active rules were analyzed, the rules were modified after the analysis. In this case, estimates for the days of 11/6, 18/6 and 25/6/2012 were analyzed.

Table 3. Activated rules in challenging days.

	11. 6. 2012	18. 6. 2012	25. 6. 2012
Rules No1	19	387	355
Rules No2	258, 259	174, 175, 178, 180	111, 112

Example of Modified Rule 355:

original:

*IF price IS very_high AND weather IS medium AND holiday IS no AND event IS no AND availability IS yes THEN weight IS **ssm***

new:

*IF price IS very_high AND weather IS medium AND holiday IS no AND event IS no AND availability IS yes THEN weight IS **medium***

The testing revealed inaccuracies in predictions, especially before holidays, when sales significantly exceeded normal values. However, this does not always apply. If the holiday falls on the weekend, the amount of sales will change only a little. If the holiday is on Monday, sales will be increased on Friday, or three days in advance. Based on the test results, part of the holiday rules were adjusted so that the highest values would be selected.

To verify rules for different weather, the number of undervalued estimates was compared to the number of overvalued estimates in the reporting period. The analysis showed that the ratio of undervalued and overvalued estimates is approximately the same for each kind of weather. For this reason, there was no need to modify the rules.

To verify the rules for major events, the results in the period of the annual pilgrimage were monitored. After evaluating the results, a total of 19 rules were modified. Repeated prediction was then performed. There has been a significant improvement in results. The deterioration of the results occurred in only 7 cases out of 36, with two differences being minimal.

After verification of all parts of the expert system, its results were compared with the commonly used inventory prediction methods.

- **Method 1**: To ensure a minimum amount of inventory, the same quantity of goods as was sold in the period under review was ordered. Of course, the existing inventory status was also taken into account and expected order has been reduced by inventory status.
- **Method 2**: A maximum limit has been set and the goods have been replenished to this maximum..
- **Method 3**: Prediction done by using the expert system created.

In the evaluation of the achieved results, two parameters were evaluated:

- Balance of unsold goods, surplus. Ideally, the balance should be close to zero.

- An unsupplied demand. This could be evaluated using historical data. In real operation, a special system for tracking unsatisfied requirements would have to be created.

Earlier research has shown that prediction systems can only be used effectively for fast-moving goods. For this reason, all three methods were tested for short-term goods, purchased in larger quantities, typically on pastries.

Table 4. Prediction results of goods sales: bread roll.

	Method 1	Method 2	Method 3
sum of positive surpluses	71138	71750	**32064**
number of underestimations	160	**33**	68

Table 4 indicates that **method 2** shows the least undervalued orders but at the expense of higher inventory. When choosing the first two ways of ordering goods, especially for fresh food, there is a high risk of depreciation leading to losses. Method 1 shows a much higher number of undervalued orders than **method 2**. This would negatively affect revenue that would be lower due to shortage of goods.

If we ordered goods according to the prediction of the expert system, we would be able to optimize both factors. Compared with **method 1**, we would reduce the number of undervalued orders by 58%; surpluses would be decreased by 55%.

Compared to Method 2, the surpluses would also fall by 55%. The number of undervalued orders would be increased, but would be still lower than in case of **method 1**. In case of other types of goods, very similar results were achieved.

📖 6.3.6

6. Conclusion

By applying an expert system for customer behavior modeling and then using it to predict commodity orders, there has been a significant improvement in inventory and savings amount. The system was designed in such a way to be easily expandable for other commodities, the fuzzy prediction rules would be easily modifiable and the remote access would be enabled via a client. Thanks to the use of real data, a testing system for individual parts of the expert system was also validated. This testing system has revealed the wrong (stereotypical) user approaches and helped to remove them.

Source: Frankeová, M., Farana, R., Formánek, I. & Walek, B. Fuzzy-Expert system for customer behavior prediction. *Advances in Intelligent Systems and Computing*. Volume 764, 2019, Pages 122-131. DOI: 10.1007/978-3-319-91189-2_13. 7th Computer Science On-line Conference, CSOC 2018; Zlin; Czech Republic; 25 April 2018 through 28 April 2018; Code 213719, ISSN: 21945357, ISBN: 978-331991188-5. Click here for full text.

6.3.7

The main advantage of the fuzzy logic used in this case study is:

- Readability of all rules and the ability to find the wrong rule.
- Possibility to reduce the number of rules to a number smaller than the number of input parameter values.

6.4 Hybrid movie recommender system

6.4.1

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. 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. This is one of the reasons why a wide range of companies and web applications have recently implemented systems analyzing users' behavior concerning 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 audiovisual 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).

Netflix implemented a recommender system in its application to decrease the number of canceled subscriptions and increase the average time of user interaction with the application (i.e., the number of streaming hours). The company anticipates that a combination of recommendations (Trending now, Continue Watch- ing, 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.

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 the favorite and unpopular genres of the user, and the final list is created using a fuzzy expert system, which evaluates movie importance.

🕮 6.4.2

2. Related work and current state of recommender systems

Recommender systems have been developed for many years and have been applied in numerous problem domains: tourism, advertisement, e-commerce, music, 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. Such an approach has also been implemented in MovieExplain. 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. 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:

- 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.

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 non-personalized or personalized recommendations. The gray-sheep problem usually arises when the organization focuses on products where the user rating is very subjective, e.g., the sale of artworks or paintings. A user who loves Leonardo da Vinci's paintings will not necessarily love his statues, but some users do. A cold visitor is concerned about 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 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) depend on two data types:

1. description and structure of their attribute and
2. user profiles generated from their feedback on various items.

The advantage of the system is its ability to solve the cold-start problem concerning 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 to extract and process of item content to calculate the similarity to other items. It is also given the ability to create a user profile based on explicit and implicit feedback. In the area of movie recommender systems, a mash-up system was published 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. Other systems and their characteristics are described in the literature review. 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 neighbour 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. Another movie recommender system uses machine learning techniques called inductive learning. This technique enables decreased sparsity and scalability in the performed experiments. 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. 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. Other systems and their characteristics are described in the literature review.

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. 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. 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 effectiveness.

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.

A simple hybrid system combining the content-based filtering approach with the collaborative filtering approach, and the user sees explanations for the recommended items is also available in the literature. 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 with recommending movies based on user emotions.

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 prediction improvement when using the hybrid approach in comparison with the traditional collaborative filtering approach or the content-based one. 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. Other hybrid systems are described in the literature review (Véras et al., 2015).

The preceding discussion and literature show 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. Fuzzy logic was successfully used in recommender systems, where user preferences and object properties are represented by fuzzy sets or in the design of the most suitable items recommended based on incomplete or certain information. Recommender systems based on fuzzy logic have been developed, e.g., in the field of e-commerce, to recommend suitable products, such as books, music, and movies, or to design suitable consumer products, such as mobile phones, tablets, and computers. A study 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 systems based on fuzzy logic, which use direct information (user rating or feedback), have been published and developed for various problem domains. 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 ex- pert system to support decision-making concerning a prefinal recommendation of items for a user.

📖 6.4.3

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 the next figure:

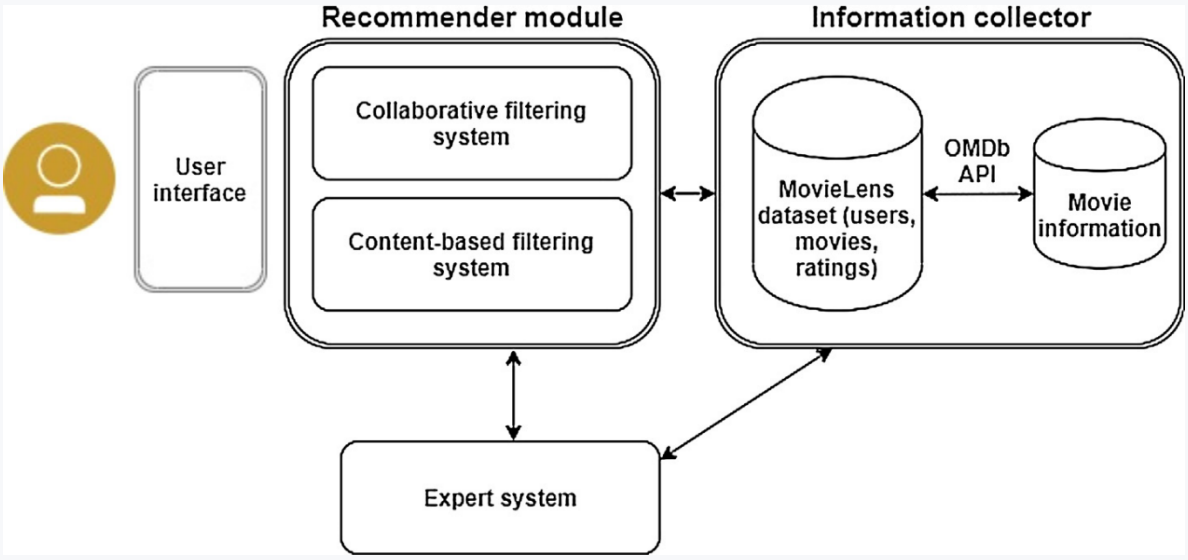


Fig. 1. Architecture of the proposed recommender system Predictory.

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 the next figure:

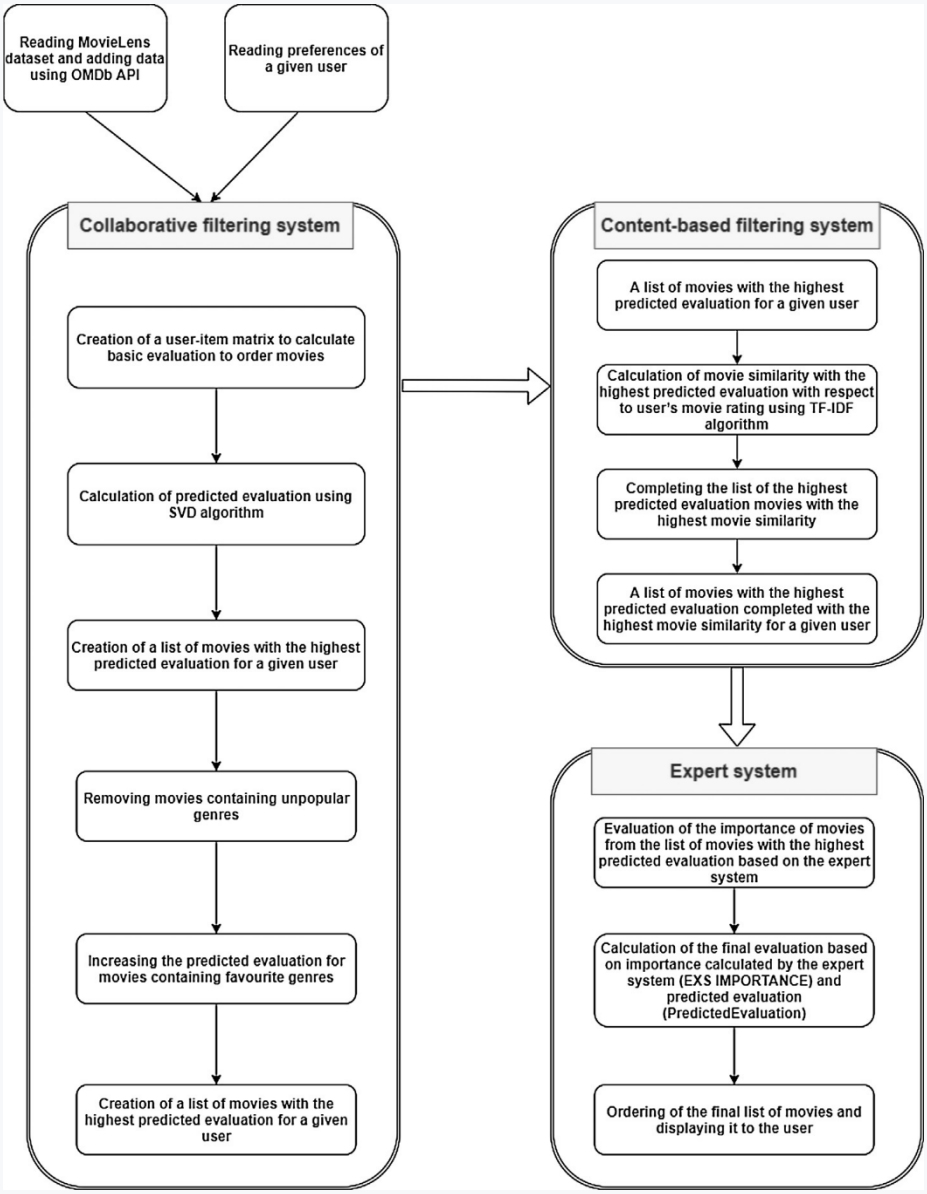


Fig. 2. Methodology of the proposed recommender system.

🕮 6.4.4

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, 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 Chapter 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 Chapter 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 the next Table.

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

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 < 0.5, 5 >, 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. 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
- Director
- Actors
- Languages
- Movie description

The method for reading the movie data is depicted in the next figure:

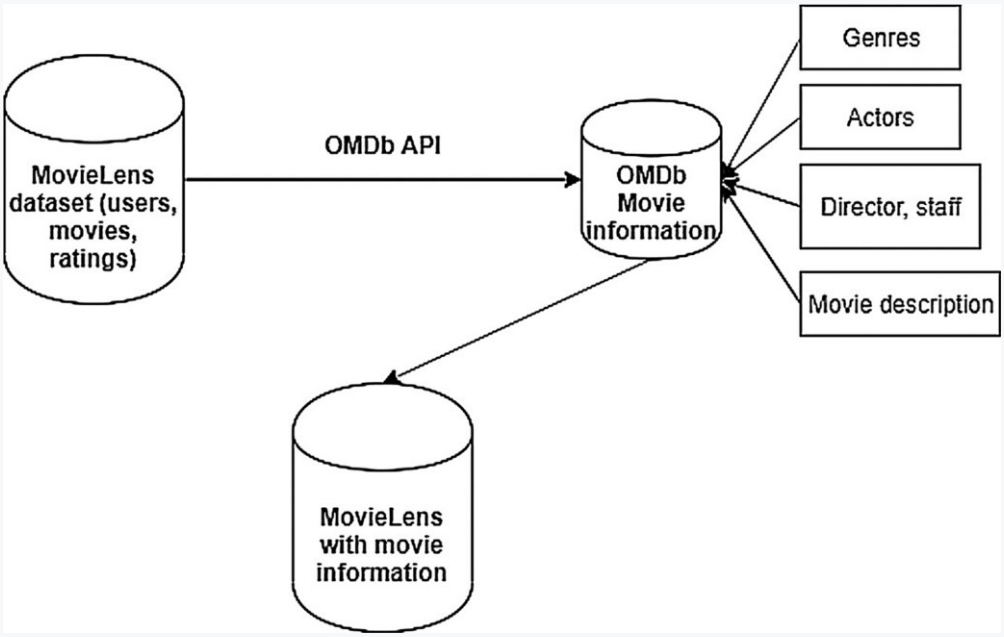


Fig. 3. Method for reading movie information from the OMDb database.

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.

🕮 6.4.5

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 the 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 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 the ratings of other users. Currently, there are two basic groups of collaborative filtering systems:

- 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 similarities to them. To choose an appropriate group for 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). 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 2 Test results for the user-based approach. Table 3 Test results for the item-based approach. Table 4 Test results for single-value decomposition.

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

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

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 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 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. 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; The 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, the item is a movie and T represents a specific rated movie

When using the SVD algorithm, Σ will always be a diagonal matrix. Rating acquisition then takes place using a scalar product U and ΣV^T matrices on a given position. The process of decomposing matrix M is depicted in the next figure:

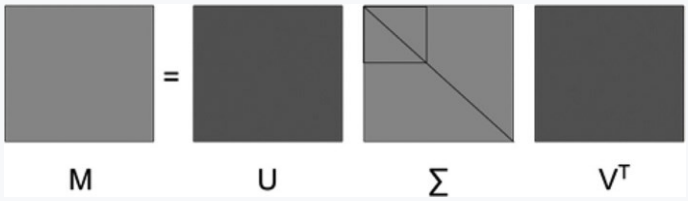


Fig. 4. Process of decomposing matrix M

The principle of the algorithm function is depicted in the next figure:

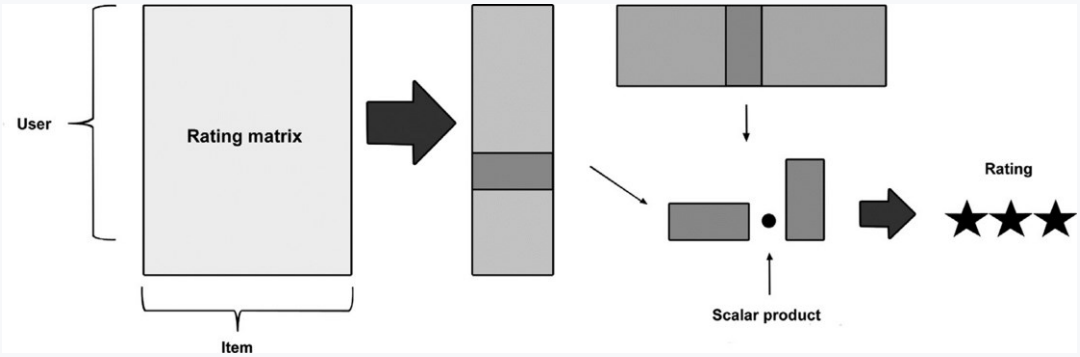


Fig. 5. Principle of the SVD algorithm function in the collaborative filtering system.

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 the next figure:

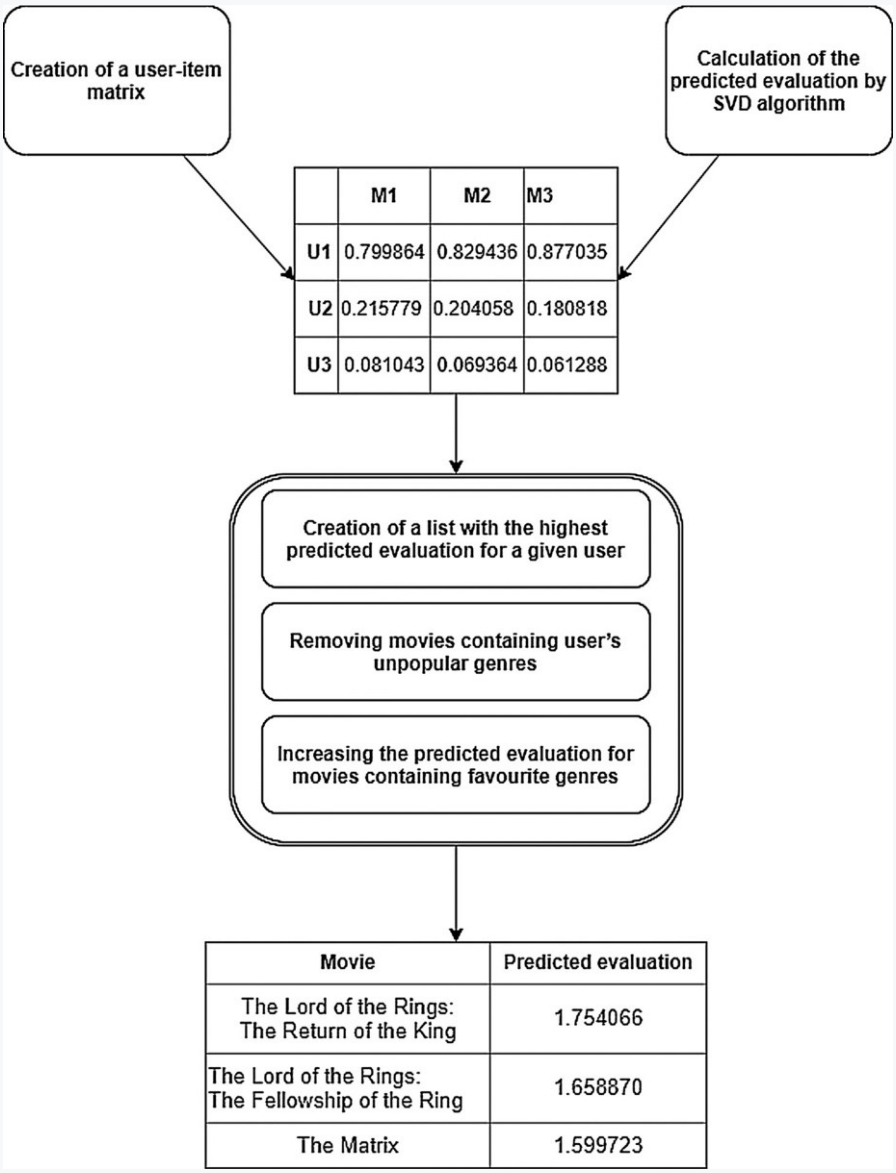


Fig. 6. Principle of collaborative filtering system functioning.

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.

Algorithm 1 Creation of the user-item matrix.

```
Input:
M represents a set of all movies in the MovieLens dataset {M1, M2, M3,..., Mn},
U represents a set of all users rating movies – users in the MovieLens dataset + other users who took part in testing the system {U1, U2, U3..., Un},
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(Ui) in Users do //For all users
  {
    foreach(Mj) 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
    }
    }
```

The next step consists of using the SVD algorithm to calculate the 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.

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

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 the user-item matrix containing the 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

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, Σ, and V^T
4. Calculate matrices U, Σ, and V^T using SVD
5. Reduce matrices U, Σ, 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 in 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 the 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.

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

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 favorites. 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.

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

🕮 6.4.6

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. 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 Co- sine 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. 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 un- necessary noise), and suffixes of individual words and creates tri-grams (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 be- tween a particular movie from a 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 high- est 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 movie
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.

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 Mj
    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 Mi and Mj
    If(SimMiMj > MaxSim)
      MaxSim = SimMiMj
    End If
  End For
M  jSim = 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

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

📖 6.4.7

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. 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:

```
IF (INP1 is low) and (INP2 is few) and (INP3 is low) THEN (IM-
PORTANCE is very low)
IF (INP1 is low) and (INP2 is few) and (INP3 is very high)
THEN (IMPORTANCE is low)
```

```
IF (INP1 is medium) and (INP2 is few) and (INP3 is very high)
THEN (IMPORTANCE is medium)
IF (INP1 is medium) and (INP2 is much) and (INP3 is very high)
THEN (IMPORTANCE is medium)
IF (INP1 is high) and (INP2 is much) and (INP3 is very high)
THEN (IMPORTANCE is high)
IF (INP1 is high) and (INP2 is very much) and (INP3 is very
high) THEN (IMPORTANCE is very high)
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.

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

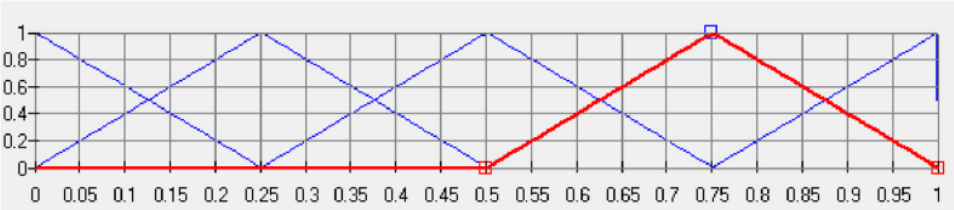
Within the inference and defuzzification, the resulting crisp number is calculated, which represents the final numerical value EXS IMPORTANCE – see Table 16.

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

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.

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



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 IM- PORTANCE, 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:

```
If ( PredictedEvaluation = < 0 ) FinalEvaluation =
PredictedEvaluation × EXS IMPORTANCE
If ( PredictedEvaluation > 0 ) FinalEvaluation =
PredictedEvaluation × ( 1 + 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 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.

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

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.

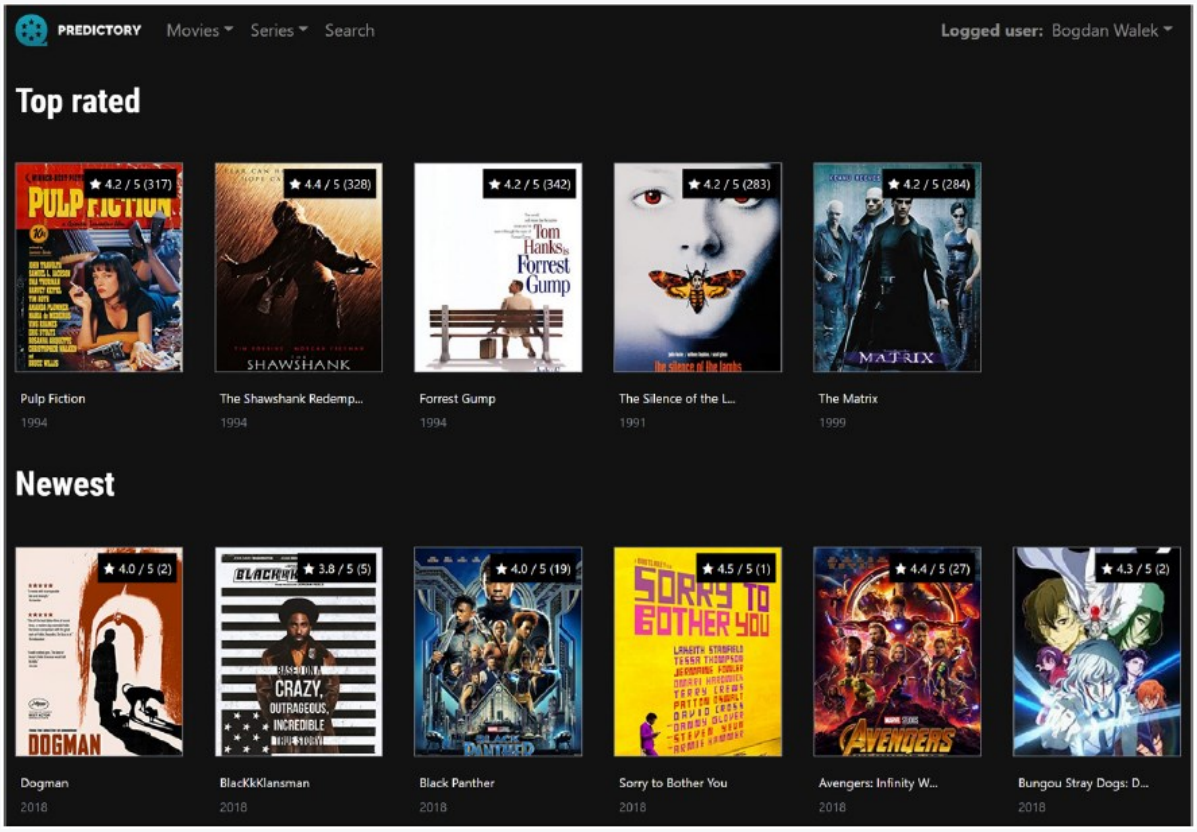


Fig. 8. Main page of the system Predictory.

📖 6.4.8

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. 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:

Precision = Correctly recommended content / 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/Rele v ant 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:

$$\mathbf{F1 - measure = 2 * precision * recall/precision + recall}$$

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, who rated a large number of movies as well as 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 a group of real users is one of the possibilities that has also been used in other research projects. 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 based on 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%**

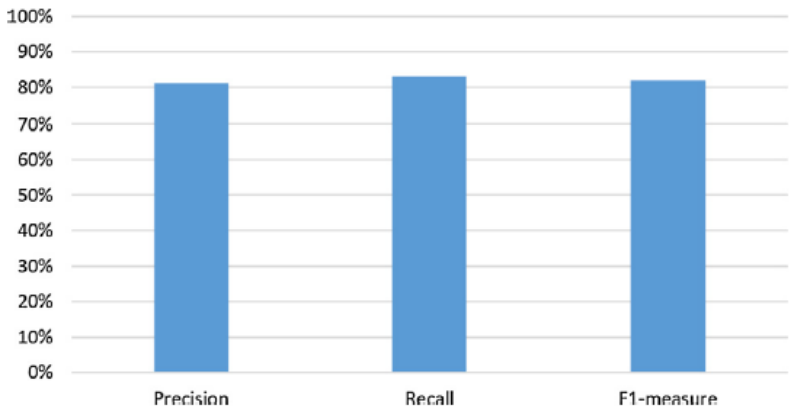


Fig. 9. Precision, recall, F1-measure metrics assessment.

📖 6.4.9

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:

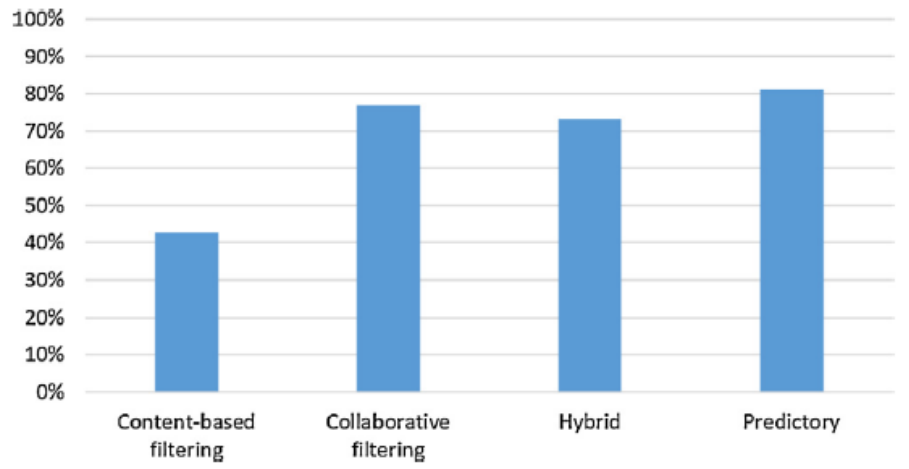
- Collaborative filtering system – a system using a pure SVD algorithm. The system was selected for its vast number of users and movie ratings, which are part of the MovieLens dataset.
- Content-based filtering system – a system using the TF-IDF algorithm. The system was selected for its large number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

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



🕮 6.4.10

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. The system is written in Python (the same as the analytical API of the proposed system) using the Django framework. The database system used in this system is PostgreSQL. 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 Graph, which enables the analysis and processing of dependencies of data, in this case, dependencies between users. 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 movie's 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 systems 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 the most relevant results, 21% of the respondents (3 out of 14) stated that MovieGeek gave the 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.

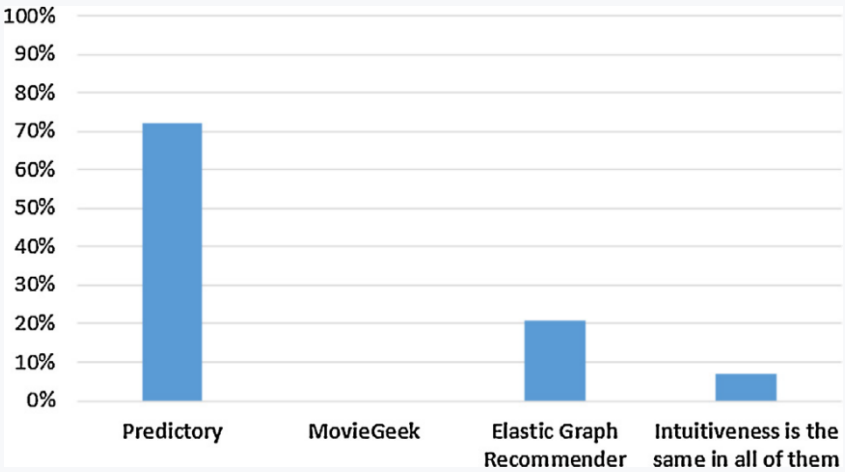


Fig. 11. Results of the comparison of the interface intuitiveness between the tested systems – Question 1.

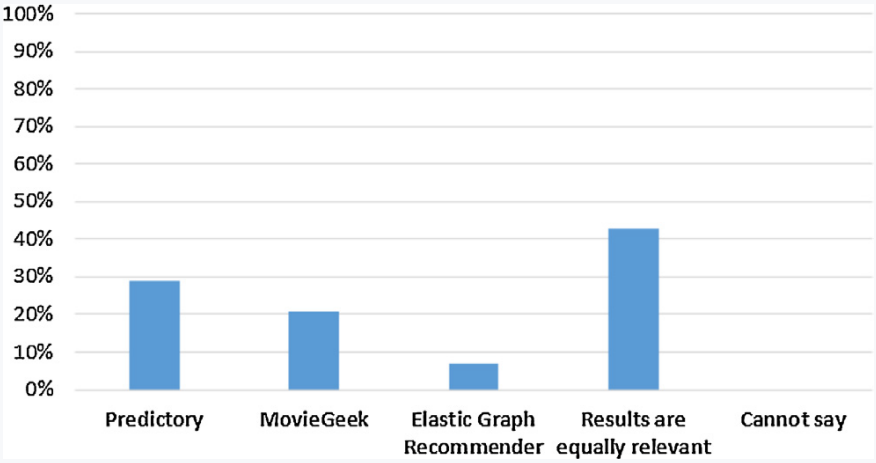


Fig. 12. Results of the comparison of the recommended movies relevance between the tested system – Question 2.

📖 6.4.11

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. Nevertheless, the area of hybrid movie recommender systems has witnessed many systems described in the literature review. 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. The E-MRS system is interesting in its ability to propose suitable movies based on emotions. However, it was verified by a small number of users. Another perspective is recommender systems with explanations of why a given movie has been recommended; an example is MoviExplain. This explanatory approach increases the credibility of the system and user loyalty. 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. The combination of collaborative filtering, content-based, and knowledge-based systems is also limited. The authors 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, re-call, 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.

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 in 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, a 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 from 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. 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.

Source: This case study was presented in collaboration with the authors with their kind permission. For more details, see:

Walek, Bogdan and Fojtik, Vladimir. A hybrid recommender system for recommending relevant movies using an expert system. Online. *Expert Systems with Applications*. 2020, year 158. ISSN 09574174. Available from Internet: https://doi.org/10.1016/j.eswa.2020.113452. [cit. 2024-02-27].

6.4.12

The results presented in this case study have several practical implications:

- The system works with various user preferences.
- 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
- It provides a functional combination of the collaborative filtering approach, a content-based approach, and a fuzzy expert system.
- Based on the results from the experimental verification, standard metrics achieved promising values.

- Proposed hybrid system using an expert system was fully implemented in a web application, Predictory

Traditional, Fuzzy and ML Systems Examples

Chapter 7

7.1 Item-based collaborative filtering

📖 7.1.1

The main purpose of this chapter is to compare traditional, fuzzy logic and traditional ML methods for movie recommender systems. Data has been selected from https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies and reduced for using in Priscilla system. You can use original dataset and run it in your environment.

📝 7.1.2

Project: Item-based collaborative filtering

(by https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/)

Dataset

- original: https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset
- local: movies - https://priscilla.fitped.eu/data/recommender_systems/movie_lens_small/movies.csv; ratings - https://priscilla.fitped.eu/data/recommender_systems/movie_lens_small/ratings.csv

The item-based collaborative filtering approach works by comparing movies to find those with similar characteristics, based on the preferences of users who have rated both movies. By evaluating the similarity scores between the selected movie and others in the system, we can recommend a list of similar movies that align with the user's preferences.

The system does this by identifying pairs of movies that have been rated or liked by the same users. It then calculates how similar the movies are based on ratings from users who rated both movies. If the ratings show significant similarity (for example, the majority of users who liked one also liked the other), the system considers these movies similar.

For example, if we compare two movies, "A" and "B", we will examine the ratings from users who rated both. If these ratings are very similar, it indicates that movies "A" and "B" share qualities that appeal to similar audiences. Therefore, if the user liked

movie "A", the system would recommend movie "B" because users tend to like both movies.

In this implementation, when a user searches for a specific movie, movie recommendation system will suggest the top 10 similar movies based on the user's interest. To achieve this, we will use an item-based collaborative filtering algorithm, which identifies movies that are similar to the one searched by the user.

1. Libraries and dataset import

```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
# import matplotlib.pyplot as plt
# import seaborn as sns
movies =
pd.read_csv("https://priscilla.fitped.eu/data/recommender_systems/movie_lens_small/movies.csv")
ratings =
pd.read_csv("https://priscilla.fitped.eu/data/recommender_systems/movie_lens_small/ratings.csv")

print(movies.head())
```

Program output:

```
   movieId                                    title  \
0        1                        Toy Story (1995)
1        2                          Jumanji (1995)
2        3                 Grumpier Old Men (1995)
3        4                Waiting to Exhale (1995)
4        5      Father of the Bride Part II (1995)

                                         genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                        Comedy|Drama|Romance
4                                       Comedy
```

The movie dataset has the following fields:

- movieId - key identifier
- title - for each movie from this dataset

- genres - are not required for this filtering approach

```
print(ratings.head())
```

Program output:

```
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```

Ratings dataset has:

- userId - unique for each user.
- movieId - to relationship with specific movie in previous dataset.
- rating - each user gives ratings for selected movies.

We can create a table containing a list of all users and all ratings.

```
final_dataset =
ratings.pivot(index='movieId',columns='userId',values='rating'
)
print(final_dataset.head())
```

Program output:

```
userId   1    2    3    4    5    6    7    8    9    10   ...
601  602  603  \
movieId                                                     ...
1        4.0  NaN  NaN  NaN  4.0  NaN  4.5  NaN  NaN  NaN  ...
4.0  NaN  4.0
2        NaN  NaN  NaN  NaN  NaN  4.0  NaN  4.0  NaN  NaN  ...
NaN  4.0  NaN
3        4.0  NaN  NaN  NaN  NaN  5.0  NaN  NaN  NaN  NaN  ...
NaN  NaN  NaN
4        NaN  NaN  NaN  NaN  NaN  3.0  NaN  NaN  NaN  NaN  ...
NaN  NaN  NaN
5        NaN  NaN  NaN  NaN  NaN  5.0  NaN  NaN  NaN  NaN  ...
NaN  NaN  NaN

userId   604  605  606  607  608  609  610
```

```
movieId
1         3.0  4.0  2.5  4.0  2.5  3.0  5.0
2         5.0  3.5  NaN  NaN  2.0  NaN  NaN
3         NaN  NaN  NaN  NaN  2.0  NaN  NaN
4         NaN  NaN  NaN  NaN  NaN  NaN  NaN
5         3.0  NaN  NaN  NaN  NaN  NaN  NaN

[5 rows x 610 columns]
```

We can observe that a user with ID 1 has rated movies 1 and 3 with a score of 4, while the other movies in their list remain unrated. To handle these unrated films effectively, we assign a rating of 0 to them. This approach allows for a smooth processing flow in the collaborative filtering algorithm, ensuring that the system can still evaluate and compare all movies, even if some remain unrated by specific users.

By filling in unrated movies with a 0 rating, we create a complete dataset matrix. This method prevents gaps in data that might otherwise hinder calculations, such as similarity scores, and keeps the algorithm consistent across all users and movies. This normalization helps our system maintain accurate recommendations while accounting for variations in user activity.

```
final_dataset.fillna(0,inplace=True)
print(final_dataset.head())
```

Program output:

```
userId   1    2    3    4    5    6    7    8    9    10   ...
601  602  603  \
movieId                                                     ...
1        4.0  0.0  0.0  0.0  4.0  0.0  4.5  0.0  0.0  0.0  ...
4.0  0.0  4.0
2        0.0  0.0  0.0  0.0  0.0  4.0  0.0  4.0  0.0  0.0  ...
0.0  4.0  0.0
3        4.0  0.0  0.0  0.0  0.0  5.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0
4        0.0  0.0  0.0  0.0  0.0  3.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0
5        0.0  0.0  0.0  0.0  0.0  5.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0

userId   604  605  606  607  608  609  610
movieId
1        3.0  4.0  2.5  4.0  2.5  3.0  5.0
2        5.0  3.5  0.0  0.0  2.0  0.0  0.0
3        0.0  0.0  0.0  0.0  2.0  0.0  0.0
```

```
4         0.0  0.0  0.0  0.0  0.0  0.0  0.0
5         3.0  0.0  0.0  0.0  0.0  0.0  0.0

[5 rows x 610 columns]
```

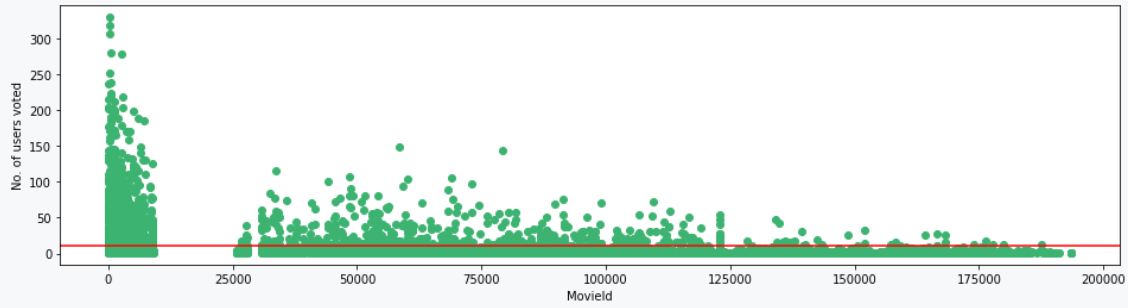
In real-world scenarios, ratings data tend to be highly sparse, as most users rate only a few movies, and these ratings are often concentrated around popular films and highly engaged users. This sparseness can introduce noise, making it challenging for a recommendation system to provide accurate predictions across all users and movies. To address this, we introduce specific filtering criteria to qualify the data, which helps reduce noise and improve recommendation quality.

- **Movie Qualification**: For a movie to be included in the final dataset, it must have received ratings from at least 10 unique users. This threshold ensures that only movies with a reasonable amount of engagement and feedback are considered, filtering out less popular or obscure films that lack sufficient data for reliable recommendations.
- **User Qualification**: To qualify as a user in the dataset, a minimum of 50 movies should have been rated by that user. By focusing on users who have rated a substantial number of movies, we reduce the influence of casual or less engaged users. This ensures that only users with meaningful interaction with the platform contribute to the recommendation process, improving the accuracy of user similarity calculations.

```
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt

# Count the number of ratings each movie has received, grouped
by 'movieId'
no_user_voted =
ratings.groupby('movieId')['rating'].agg('count')

# Count the number of movies each user has rated, grouped by
'userId'
no_movies_voted =
ratings.groupby('userId')['rating'].agg('count')

# Set up a figure and axis for the plot with a specific size
f, ax = plt.subplots(1, 1, figsize=(16, 4))

# Scatter plot showing the number of users who rated each
movie by 'movieId'
```

```
plt.scatter(no_user_voted.index, no_user_voted,
color='mediumseagreen')

# Draw a horizontal red line at y=10 to indicate the threshold
of 10 user votes for movie qualification
plt.axhline(y=10, color='r')

# Label the x-axis as 'MovieId' and the y-axis as 'No. of
users voted'
plt.xlabel('MovieId')
plt.ylabel('No. of users voted')

# Display the plot
plt.show()
```

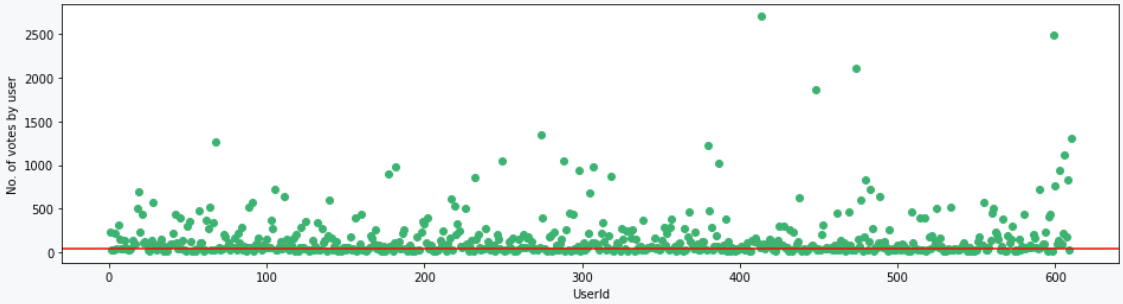
Program output:




```
# Filter 'final_dataset' to include only movies that have been
rated by more than 10 users.
# 'no_user_voted > 10' filters the movies with more than 10
votes
# '.index' retrieves the indices (movie IDs) that satisfy this
condition
final_dataset = final_dataset.loc[no_user_voted[no_user_voted
> 10].index, :]

# Set up a matplotlib figure with specific size for plotting
f, ax = plt.subplots(1, 1, figsize=(16, 4))

# Plot the number of movies each user has rated
(no_movies_voted) as scatter plot
plt.scatter(no_movies_voted.index, no_movies_voted,
color='mediumseagreen')

# Add a horizontal line at y = 50 to indicate the minimum vote
threshold for users
```

```
plt.axhline(y=50, color='r')

# Label the x-axis as 'UserId' and the y-axis as 'No. of votes
by user'
plt.xlabel('UserId')
plt.ylabel('No. of votes by user')

# Display the plot
plt.show()
```

Program output:




```
# Filter 'final_dataset' to include only users who have rated
more than 50 movies.
# 'no_movies_voted > 50' filters the users with more than 50
ratings.
# '.index' retrieves the indices (user IDs) that satisfy this
condition.
final_dataset = final_dataset.loc[:,
no_movies_voted[no_movies_voted > 50].index]

# Display the filtered dataset (final_dataset).
print(final_dataset)
```

Program output:

```
userId   1    4    6    7    10   11   15   16   17   18   ...
600  601  602  \
movieId                                                      ...
1        4.0  0.0  0.0  4.5  0.0  0.0  2.5  0.0  4.5  3.5  ...
2.5  4.0  0.0
2        0.0  0.0  4.0  0.0  0.0  0.0  0.0  0.0  0.0  3.0  ...
4.0  0.0  4.0
3        4.0  0.0  5.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0
5        0.0  0.0  5.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
2.5  0.0  0.0
```

```
6         4.0  0.0  4.0  0.0  0.0  5.0  0.0  0.0  0.0  4.0  ...
0.0  0.0  3.0
...       ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
...  ...  ...
174055    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  4.0  0.0
176371    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  4.0  0.0
177765    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  4.5  0.0
179819    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0
187593    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...
0.0  0.0  0.0

userId    603  604  605  606  607  608  610
movieId
1         4.0  3.0  4.0  2.5  4.0  2.5  5.0
2         0.0  5.0  3.5  0.0  0.0  2.0  0.0
3         0.0  0.0  0.0  0.0  0.0  2.0  0.0
5         0.0  3.0  0.0  0.0  0.0  0.0  0.0
6         4.0  3.0  0.0  0.0  0.0  0.0  5.0
...       ...  ...  ...  ...  ...  ...  ...
174055    0.0  0.0  0.0  0.0  0.0  0.0  0.0
176371    0.0  0.0  0.0  0.0  0.0  0.0  0.0
177765    0.0  0.0  0.0  0.0  0.0  0.0  0.0
179819    0.0  0.0  0.0  0.0  0.0  0.0  0.0
187593    0.0  0.0  0.0  0.0  0.0  0.0  0.0

[2121 rows x 378 columns]
```

2. Preprocessing sparse data

Our final_dataset has dimensions of 2121 * 378. This matrix is mostly sparse, meaning that most of the values are zeros (indicating no rating for a particular movie by a user).

A **Compressed Sparse Row** (CSR) matrix is used to store sparse data efficiently by only keeping non-zero values, which reduces memory usage significantly. This format is particularly useful when dealing with large datasets like the one in the example, where a lot of ratings are missing (or are zero).

The csr_matrix function from the scipy.sparse library is used to convert the dense matrix (final_dataset) into the CSR format. This conversion reduces the memory

footprint and speeds up operations like matrix multiplication or finding non-zero values.

```
from scipy.sparse import csr_matrix

# Convert the 'final_dataset' DataFrame values to a compressed sparse row (CSR) matrix.
# This format is efficient for storing sparse data, where most elements are zero.
# 'csr_data' will be used for faster similarity computations in our recommendation system.
csr_data = csr_matrix(final_dataset.values)

# Display the sparse matrix (only non-zero elements are stored efficiently).
print(csr_data)
```

Program output:

```
  (0, 0)     1.0
  (0, 1)     4.0
  (0, 4)     4.5
  (0, 7)     2.5
  (0, 9)     4.5
  (0, 10)    3.5
  (0, 11)    4.0
  (0, 13)    3.5
  (0, 17)    3.0
  (0, 20)    3.0
  (0, 21)    3.0
  (0, 26)    5.0
  (0, 29)    5.0
  (0, 30)    4.0
  (0, 32)    3.0
  (0, 35)    5.0
  (0, 39)    5.0
  (0, 40)    4.0
  (0, 41)    4.0
  (0, 42)    2.5
  (0, 44)    4.5
  (0, 47)    0.5
  (0, 48)    4.0
  (0, 51)    2.5
  (0, 54)    4.0
  :    :
```

```
  (2118, 358)       4.0
  (2118, 370)       4.5
  (2119, 0)  179819.0
  (2119, 38) 3.5
  (2119, 63) 3.0
  (2119, 99) 0.5
  (2119, 128)       4.5
  (2119, 157)       4.5
  (2119, 237)       0.5
  (2119, 257)       4.5
  (2119, 318)       2.0
  (2119, 346)       2.0
  (2119, 358)       5.0
  (2119, 366)       3.5
  (2120, 0)  187593.0
  (2120, 38) 4.0
  (2120, 63) 5.0
  (2120, 147)       2.5
  (2120, 156)       4.5
  (2120, 157)       5.0
  (2120, 187)       5.0
  (2120, 206)       4.0
  (2120, 237)       3.0
  (2120, 318)       3.5
  (2120, 358)       4.0
```

There is no sparse value in the csr_sample, and values are assigned as rows and column indexes. For the 0th row and 4nd column, the value is 4.5. Applying the csr_matrix method to the dataset:

```
# Reset the index of 'final_dataset' to ensure it has a
default integer index.
# This helps with direct indexing when we need to reference
rows by position
# in the recommendation system.
final_dataset.reset_index(inplace=True)
```

3. Model training

Now we need to find the similarity between the items. The goal of this step is to find items that are most similar to the user's previous preferences or other items that the user has liked or highly rated. As a result, the system can provide personalized recommendations by identifying items that are close to the user's existing tastes or that match the patterns of similar users.

In collaborative filtering, the system must quantify the proximity of users or items in order to recommend relevant items. Using the similarity metric, we can thus create a ranked list of items that are most similar to what the user has already interacted with. For example, if a user likes "Movie A," the system can find other movies with high similarity scores to "Movie A" and recommend them as potential next choices. Metrics ensure that recommendations are based on patterns of similarity, not just random selections, making them more relevant to the user.

We use cosine distance metric which is very fast and preferable. Cosine distance measures the angle between two vectors, which is useful when comparing the orientation or direction of two vectors in multi-dimensional space. In a recommendation system, cosine distance is effective when comparing items (like movies) because it doesn't take into account the magnitude of the vectors (ratings), only their direction or pattern. If two users have the same preferences but different rating scales, cosine distance will still recognize them as similar, making it preferable over other distance metrics like Euclidean or Pearson when working with sparse data (e.g., missing ratings).

```
# Importing the necessary class for nearest neighbors search
from sklearn.neighbors import NearestNeighbors

# Initializing the NearestNeighbors model with the cosine
distance metric
# metric='cosine' specifies that the distance between vectors
will be calculated using the cosine similarity metric.
# algorithm='brute' indicates that we will use brute force
(exhaustive search) to find the nearest neighbors.
# n_neighbors=20 means we are looking for the 20 nearest
neighbors for each point.
# n_jobs=-1 allows parallel computation using all available
CPU cores for faster processing.
knn = NearestNeighbors(metric='cosine', algorithm='brute',
n_neighbors=20, n_jobs=-1)

# Fitting the model on the data. Here 'csr_data' is assumed to
be a sparse matrix (likely of type csr_matrix).
# The NearestNeighbors model will learn the structure of the
dataset and prepare it for fast nearest neighbor search.
knn.fit(csr_data)

# The final model setup is displayed below with additional
parameters like leaf_size, which defines the size of leaf
nodes in the tree.
# The p=2 is used for the Minkowski distance, which is the
generalization of Euclidean distance.
```

```
# NearestNeighbors(algorithm='brute', leaf_size=30,
metric='cosine',
#                  metric_params=None, n_jobs=-1,
n_neighbors=20, p=2, radius=1.0)
print('done')
```

Program output:

```
done
```

4. Making the recommendation function

The get_movie_recommendation function works by taking a movie title as input, identifying similar movies, and returning the top 10 recommendations. It works as follows:

- The function first checks if the input movie title matches any titles in the movies dataset. If a match is found, the movieId of the first match is retrieved.
- Using the movie ID, the function searches for that movie in the final_dataset and then calculates the similarity distance between that movie and all other movies. The neighbors function (from knn) is used to obtain indices and similarity scores for the nearest movies (based on the cosine metric).
- The function then selects and ranks the closest movies based on their similarity, excluding the input movie itself.
- For each recommended movie, the function collects its title and distance and stores them in a list. This list is finally transformed into a DataFrame that displays the top 10 recommended movies and their similarity scores.

```
def get_movie_recommendation(movie_name):
    # Set the number of recommendations to be returned
    n_movies_to_reccomend = 10

    # Step 1: Find the movie(s) with titles matching the input
name in the 'movies' dataset
    movie_list =
movies[movies['title'].str.contains(movie_name)]

    # Check if any movie was found
    if len(movie_list):
        # Retrieve the movieId of the first matching movie and
its index in final_dataset
        movie_idx = movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] ==
movie_idx].index[0]
```

```
        # Step 2: Get the distances and indices of the top
similar movies
        distances, indices =
knn.kneighbors(csr_data[movie_idx],
n_neighbors=n_movies_to_reccomend + 1)

        # Step 3: Sort recommended movies by distance,
excluding the input movie itself
        rec_movie_indices = sorted(
            list(zip(indices.squeeze().tolist(),
distances.squeeze().tolist())),
            key=lambda x: x[1]
        )[:0:-1]

        # Prepare a list to store recommended movie details
        recommend_frame = []

        # Step 4: Retrieve and format titles and distances for
the top recommended movies
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
            recommend_frame.append({'Title':
movies.iloc[idx]['title'].values[0], 'Distance': val[1]})

        # Step 5: Convert the recommendations list to a
DataFrame and return it
        df = pd.DataFrame(recommend_frame, index=range(1,
n_movies_to_reccomend + 1))
        return df

    # If no movie matches, prompt user to check input
    else:
        return "No movies found. Please check your input"
```

5. Recommendation

- Finally, we can try to recommend some movies...

```
print(get_movie_recommendation('Iron Man'))
```

Program output:

```
Index(['movieId', 'title', 'genres'], dtype='object')
```

```
                              Title      Distance
1         X-Men: First Class (2011)  1.111104e-07
2   Guardians of the Galaxy (2014)  1.106718e-07
3                 District 9 (2009)  1.085074e-07
4            Sherlock Holmes (2009)  1.077268e-07
5              Kung Fu Panda (2008)  1.066761e-07
6                   Watchmen (2009)  1.061401e-07
7                  Star Trek (2009)  1.047793e-07
8                 Iron Man 2 (2010)  9.618056e-08
9                     Avatar (2009)  9.313901e-08
10             Avengers, The (2012)  8.755834e-08
```

or another:

```
print(get_movie_recommendation('Memento'))
```

Program output:

```
                                                    Title
Distance
1                                      Spider-Man (2002)
0.000047
2   Pirates of the Caribbean: The Curse of the Bla...
0.000047
3                                   Donnie Darko (2001)
0.000047
4                             Kill Bill: Vol. 2 (2004)
0.000047
5                               Minority Report (2002)
0.000047
6   Lord of the Rings: The Return of the King, The...
0.000043
7        Eternal Sunshine of the Spotless Mind (2004)
0.000042
8                             Kill Bill: Vol. 1 (2003)
0.000042
9       Lord of the Rings: The Two Towers, The (2002)
0.000041
10  Lord of the Rings: The Fellowship of the Ring,...
0.000041
```

7.1.3

In Item-Based Collaborative Filtering, recommendations are generated based on:

- Finding similar items
- Finding similar users
- Predicting user similarity
- Random item selection

7.1.4

Which of the following are true about using the Compressed Sparse Row (CSR) format?

- CSR reduces storage by compressing row data efficiently
- CSR format allows fast row slicing in large, sparse datasets
- CSR format can store dense data structures more effectively
- CSR is slower to access non-zero elements compared to dense matrices

7.1.5

Why is it essential to calculate similarity between items in a recommendation system?

- To find items with matching features for better recommendations
- To generate recommendations based on items with a similar user base
- To randomly rank items in the dataset
- To minimize dataset sparsity

7.2 Fuzzy logic application

7.2.1

Project: Fuzzy logic expert system

Develop a fuzzy-logic expert system-based recommender system that uses fuzzy logic and expert rules to make personalized recommendations to users. Develop a comparative recommender system using conventional machine learning techniques such as collaborative filtering, content-based filtering, or hybrid techniques. Evaluate the performance of the two systems using common evaluation metrics. Compare the performance of the fuzzy-logic expert system-based recommender system with the

comparative recommender system in terms of their effectiveness, efficiency, and scalability.

Dataset

- original: https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies
- reduced local: https://priscilla.fitped.eu/data/recommender_systems/movies_dataset.csv

1. Import libraries and dataset

```
## Import neccessary libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

data =
pd.read_csv('https://priscilla.fitped.eu/data/recommender_systems/movies_dataset.csv')
print(data.head())
```

Program output:

```
       id                                title  \
0  615656                 Meg 2: The Trench
1  758323               The Pope's Exorcist
2  667538  Transformers: Rise of the Beasts
3  640146  Ant-Man and the Wasp: Quantumania
4  677179                          Creed III

                          genres original_language  \
0    Action-Science Fiction-Horror                en
1          Horror-Mystery-Thriller                en
2  Action-Adventure-Science Fiction                en
3  Action-Adventure-Science Fiction                en
4                  Drama-Action                en

                                            overview
popularity  \
0  An exploratory dive into the deepest depths of...
8763.998
```

```
1  Father Gabriele Amorth Chief Exorcist of the V...
5953.227
2  When a new threat capable of destroying the en...
5409.104
3  Super-Hero partners Scott Lang and Hope van Dy...
4425.387
4  After dominating the boxing world Adonis Creed...
3994.342

                                production_companies
release_date     budget  \
0  Apelles Entertainment-Warner Bros. Pictures-di...
02/08/2023  129000000
1  Screen Gems-2.0 Entertainment-Jesus & Mary-Wor...
05/04/2023   18000000
2  Skydance-Paramount-di Bonaventura Pictures-Bay...
06/06/2023  200000000
3             Marvel Studios-Kevin Feige Productions
15/02/2023  200000000
4  Metro-Goldwyn-Mayer-Proximity Media-Balboa Pro...
01/03/2023   75000000

     revenue  runtime    status  \
0  352056482    116.0  Released
1   65675816    103.0  Released
2  407045464    127.0  Released
3  475766228    125.0  Released
4  269000000    116.0  Released

                                             tagline
vote_average  \
0                                  Back for seconds.
7.079
1  Inspired by the actual files of Father Gabriel...
7.433
2                                     Unite or fall.
7.340
3            Witness the beginning of a new dynasty.
6.507
4                     You can't run from your past.
7.262

   vote_count
credits  \
```

```
0        1365  Jason Statham-Wu Jing-Shuya Sophia Cai-Sergio
...
1         545  Russell Crowe-Daniel Zovatto-Alex Essoe-
Franco...
2        1007  Anthony Ramos-Dominique Fishback-Luna Lauren
V...
3        2811  Paul Rudd-Evangeline Lilly-Jonathan Majors-
Kat...
4        1129  Michael B. Jordan-Tessa Thompson-Jonathan
Majo...

                                              keywords  \
0                based on novel or book-sequel-kaiju
1  spain-rome italy-vatican-pope-pig-possession-c...
2  peru-alien-end of the world-based on cartoon-b...
3  hero-ant-sequel-superhero-based on comic-famil...
4  philadelphia pennsylvania-husband wife relatio...

                         poster_path
backdrop_path  \
0  /4m1Au3YkjqsxF8iwQy0fPYSxE0h.jpg
/qlxy8yo5bcgUw2KAmmojUKp4rHd.jpg
1  /9JBEPLTPSm0d1mbEcLxULjJq9Eh.jpg
/hiHGRbyTcbZoLsYYkO4QiCLYe34.jpg
2  /gPbM0MK8CP8A174rmUwGsADNYKD.jpg
/woJbg7ZqidhpvqFGGMRhWQNoxwa.jpg
3  /qnqGbB22YJ7dSs4o6M7exTpNxPz.jpg
/m8JTwHFwX7I7JY5fPe4SjqejWag.jpg
4  /cvsXj3I9Q2iyyIo95AecSd1tad7.jpg
/5i6SjyDbDWqyun8klUuCxrlFbyw.jpg

                                       recommendations
0  1006462-298618-569094-1061181-346698-1076487-6...
1  713704-296271-502356-1076605-1084225-1008005-9...
2  496450-569094-298618-385687-877100-598331-4628...
3  823999-676841-868759-734048-267805-965839-1033...
4  965839-267805-943822-842942-1035806-823999-107...
```

2. Data preprocessing

Some preprocessing of source data must be realized.

- **Select data** whose vote count is greater than or equal to 350.
- **Drop duplicates** from title and released date.

- **Calculate** the recommendation score from the data itself.
- **Define Weights:** Three factors are considered for the recommendation score: popularity, vote_average, and vote_count. We have assigned weights to these factors to indicate their importance in the recommendation score. In this case, we've assigned/defined the following weights:
- weight_popularity with a weight of 0.4.
- weight_vote_average with a weight of 0.3.
- weight_vote_count with a weight of 0.3.
- **Calculate Recommendation Score:** The recommendation score for each movie is calculated using the provided weights and the values of the three factors. The formula for the recommendation score is as follows:

recommendation_score = (popularity * weight_popularity) + (vote_average * weight_vote_average) + (vote_count * weight_vote_count)

- **Scale the Recommendation Score:** To ensure that the recommendation scores fall within a standardized range of [0, 1], a MinMaxScaler is applied to the calculated recommendation scores. The MinMaxScaler scales the data so that the minimum value is mapped to 0, and the maximum value is mapped to 1. This scaling allows for easy comparison and interpretation of the recommendation scores across different movies.

In summary, the weights for different movie features calculate a recommendation score for each movie using these weights and then scale the recommendation scores to a standardized range to make them more interpretable and suitable for ranking or recommendations. This process helps in assessing and comparing movies based on their overall recommendation scores, considering their popularity, vote average, and vote count.

```
data.drop_duplicates(subset=["title","release_date"],
inplace=True)
data=data[data.vote_count >= 350]

from sklearn.preprocessing import MinMaxScaler,StandardScaler

# Define the weights
weight_popularity = 0.4
weight_vote_average = 0.3
weight_vote_count = 0.3
```

```
# Calculate the recommendation score
data['recommendation_score'] = (
    (data['popularity'] * weight_popularity) +
    (data['vote_average'] * weight_vote_average) +
    (data['vote_count'] * weight_vote_count)
)

# Create a MinMaxScaler
scaler = MinMaxScaler()

# Scale the recommendation score to the range [0, 1]
data['recommendation_score'] =
scaler.fit_transform(data['recommendation_score'].values.resha
pe(-1, 1))

print(data['recommendation_score'])
```

Program output:

```
0        0.384563
1        0.246224
2        0.238240
3        0.253128
4        0.184798
           ...
38220    0.000000
38611    0.096083
38670    0.000643
38732    0.000791
50393    0.048203
Name: recommendation_score, Length: 7651, dtype: float64
```

The 'genres' column in the dataset is transformed into a set of binary columns, one for each genre category, following these steps:

1. **Handling Missing Values:** Initially, any missing values in the 'genres' column are replaced with empty strings to ensure consistency and prevent errors during further processing.
2. **Creating Binary Columns:** The `str.get_dummies` function is applied to the 'genres' column. This function splits the hyphen-separated genre categories and creates binary columns for each unique genre. For example, if a movie belongs to the 'Action' and 'Adventure' genres, two binary columns will be created, one for 'Action' and another for 'Adventure'. If the movie does not have a specific genre, the corresponding column will contain a 0.

3. **Concatenating Binary Columns:** The binary columns created in the previous step are concatenated with the original DataFrame. This means that for each movie, you now have a set of binary columns indicating its genre affiliations.
4. **Dropping Original 'genres' Column:** The original 'genres' column, which contained the hyphen-separated genre categories, is no longer needed since the information is now represented in binary format. It is dropped from the DataFrame to keep it clean and prevent redundancy.
5. **Handling Remaining Missing Values:** In the last step, any remaining missing values in the DataFrame (which might occur if other columns had missing values) are filled with 0. This ensures that the binary genre columns do not have missing values, making them suitable for further analysis or modeling.

In summary, this code efficiently transforms the categorical 'genres' information into a set of binary columns representing each genre category, making it easier to work with and analyze genre-related aspects of the dataset

```
# Step 1: Handle Missing Values
# Replace any missing values in the 'genres' column with empty
strings
data['genres'] = data['genres'].fillna('')

# Step 2: Create Binary Columns
# Use str.get_dummies to split the genres and create binary
(one-hot encoded) columns for each unique genre
genres_dummies = data['genres'].str.get_dummies(sep='-')

# Step 3: Concatenate Binary Columns
# Concatenate the newly created binary genre columns with the
original DataFrame
data = pd.concat([data, genres_dummies], axis=1)

# Step 4: Drop the Original 'genres' Column
# Remove the original 'genres' column as it's no longer needed
data = data.drop('genres', axis=1)

# Step 5: Handle Remaining Missing Values
# Fill any remaining missing values in the DataFrame with 0
data = data.fillna(0)

# Display the transformed DataFrame
print(data.head())
```

Program output:

```
       id                                title original_language
\
0  615656                   Meg 2: The Trench                en
1  758323                The Pope's Exorcist                 en
2  667538   Transformers: Rise of the Beasts                 en
3  640146  Ant-Man and the Wasp: Quantumania                 en
4  677179                           Creed III                en

                                             overview
popularity  \
0  An exploratory dive into the deepest depths of...
8763.998
1  Father Gabriele Amorth Chief Exorcist of the V...
5953.227
2  When a new threat capable of destroying the en...
5409.104
3  Super-Hero partners Scott Lang and Hope van Dy...
4425.387
4  After dominating the boxing world Adonis Creed...
3994.342

                                production_companies
release_date     budget  \
0  Apelles Entertainment-Warner Bros. Pictures-di...
02/08/2023  129000000
1  Screen Gems-2.0 Entertainment-Jesus & Mary-Wor...
05/04/2023   18000000
2  Skydance-Paramount-di Bonaventura Pictures-Bay...
06/06/2023  200000000
3             Marvel Studios-Kevin Feige Productions
15/02/2023  200000000
4  Metro-Goldwyn-Mayer-Proximity Media-Balboa Pro...
01/03/2023   75000000

     revenue  runtime  ... History Horror  Music  Mystery
Romance  \
0  352056482    116.0  ...       0      1      0        0
0
1   65675816    103.0  ...       0      1      0        1
0
2  407045464    127.0  ...       0      0      0        0
0
```

```
3  475766228    125.0  ...        0       0       0         0
0
4  269000000    116.0  ...        0       0       0         0
0

  Science Fiction TV Movie Thriller War  Western
0               1        0        0   0        0
1               0        0        1   0        0
2               1        0        0   0        0
3               1        0        0   0        0
4               0        0        0   0        0

[5 rows x 39 columns]
```

3. Fuzzy part

Fuzzy logic is designed to make movie recommendations based on a user's preference for a specific movie. Here's how it works:

1. **Antecedent and Consequent Definitions:** The code utilizes the scikit-fuzzy library to create two main components: Antecedent and Consequent. The Antecedent, named "User_Preference," represents the user's preference for a movie, and the Consequent, named "Recommendation," represents the recommendation score for movies.
2. **Fuzzy Sets and Membership Functions:** For the "User_Preference" and "Recommendation" components, fuzzy sets and membership functions are defined. These functions capture the fuzzy logic by dividing the range of preferences and recommendations into linguistic categories. For example, "Love" and "Hate" represent extreme user preferences, while "Somewhat Recommended" represents a moderate recommendation.
3. **Fuzzy Rules:** Fuzzy rules are established to determine how user preferences relate to movie recommendations. In this case, three rules are defined. If a user "Loves" a movie, it will be "Highly Recommended." If they have a "Neutral" preference, it will be "Somewhat Recommended." If they "Hate" it, the movie will be "Not Recommended."
4. **Control System and Simulation:** A control system is created to manage the fuzzy logic rules, which are then used to make movie recommendations. A simulation is also created to execute the control system.
5. **User Input and Recommendation:** When a movie title and user preference are provided as input, the code simulates the recommendation process. It computes a recommendation score based on the user's preference and sorts the movie dataset by this score. The top five recommended movies, along with their recommendation scores, are returned to the user.
6. **Handling Unavailable Movies:** If the specified movie title is not found in the dataset, the code handles this case and returns a message indicating that the movie is not in the dataset.

In summary, a fuzzy logic-based movie recommendation system where user preferences are mapped to movie recommendations using fuzzy sets, rules, and membership functions. It provides users with personalized movie recommendations based on their preferences, allowing them to discover movies tailored to their liking.

```
import pandas as pd
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

```
# Create a function to generate movie recommendations based on
user preferences
def recommend_movie(movie_title, df):
    # Create Antecedent and Consequent objects
    user_preference = ctrl.Antecedent(np.arange(-1, 1.1, 0.1),
'User_Preference')
    recommendation = ctrl.Consequent(np.arange(0, 1.1, 0.1),
'Recommendation')

    # Define fuzzy sets and membership functions for user
preferences
    user_preference['Hate'] =
fuzz.trimf(user_preference.universe, [-1, -1, 0])
    user_preference['Neutral'] =
fuzz.trimf(user_preference.universe, [-1, 0, 1])
    user_preference['Love'] =
fuzz.trimf(user_preference.universe, [0, 1, 1])

    # Define fuzzy sets and membership functions for
recommendations
    recommendation['Not Recommended'] =
fuzz.trimf(recommendation.universe, [0, 0, 0.5])
    recommendation['Somewhat Recommended'] =
fuzz.trimf(recommendation.universe, [0, 0.5, 1])
    recommendation['Highly Recommended'] =
fuzz.trimf(recommendation.universe, [0.5, 1, 1])

    # Define the fuzzy rules
    rule1 = ctrl.Rule(user_preference['Love'],
recommendation['Highly Recommended'])
    rule2 = ctrl.Rule(user_preference['Neutral'],
recommendation['Somewhat Recommended'])
```

```
    rule3 = ctrl.Rule(user_preference['Hate'],
recommendation['Not Recommended'])

    # Create the control system
    recommendation_ctrl = ctrl.ControlSystem([rule1, rule2,
rule3])

    # Create a simulation
    recommendation_sim =
ctrl.ControlSystemSimulation(recommendation_ctrl)

    # Get the row for the specified movie title
    movie_row = df[df['title'] == movie_title]

    if not movie_row.empty:
        # Set the user's preference (you can adjust this
value)
        user_preference_value = 0.6

        recommendation_sim.input['User_Preference'] =
user_preference_value
        recommendation_sim.compute()

        # Get the recommendation score
        recommendation_score =
recommendation_sim.output['Recommendation']

        # Sort the DataFrame by the recommendation score
        recommended_movies = df.copy()
        recommended_movies['Recommendation_Score'] =
recommendation_score
        recommended_movies =
recommended_movies.sort_values(by='Recommendation_Score',
ascending=False)
        top_recommended_movies = recommended_movies.head(5)

        # Return the recommended movies
        return top_recommended_movies[['title',
'Recommendation_Score']]

    else:
        return "Movie not found in the dataset"
```

4. Recommendation

- Finally, we can try to recommend some movies...

```
# Example: Get movie recommendations based on a movie title
movie_title = "Black Adam"  # Replace with the actual movie
title
recommended_movies = recommend_movie(movie_title, data)

# Display the recommended movies with their recommendation
scores
print(recommended_movies)
```

Program output:

```
                         title  Recommendation_Score
0           Meg 2: The Trench              0.587805
9342  The Secret Life of Pets              0.587805
9368    A History of Violence              0.587805
9367        Nights of Cabiria              0.587805
9365            Young Guns II              0.587805
```

Previous recommendation was based on their recommendation scores...

4a. Recommendation based on title similarity

This code uses fuzzy string matching to recommend movies with titles similar to the given input title. By calculating a similarity score between the input title and all other movie titles, it identifies and returns the most similar matches. This approach is useful in cases where users may remember partial or approximate names, allowing recommendations based on fuzzy matches rather than exact ones.

- Calculate Similarity: The calculate_similarity function uses fuzzy matching to calculate a similarity score between two movie titles based on the order of the tokens.
- Generate recommendations: The recommend_similar_movies function iterates over all movie titles in the dataset, calculates a similarity score to the input title, and ranks them in descending order.
- Return top results: The function returns the first N recommendations based on the highest similarity score.

```
from fuzzywuzzy import fuzz  # Import for calculating
similarity scores
from fuzzywuzzy import process
```

```
# Function to calculate similarity between two movie titles
def calculate_similarity(movie1, movie2):
    # Calculate the similarity score based on token order,
ignoring case
    title_similarity = fuzz.token_sort_ratio(movie1, movie2)
    return title_similarity

# Function to recommend similar movies based on title
similarity
def recommend_similar_movies(movie_title,
num_recommendations=5):
    similarity_scores = []  # List to store similarity scores
with titles

    # Iterate through each title in the DataFrame to calculate
similarity
    for title in data['title']:
        similarity = calculate_similarity(movie_title, title)
        similarity_scores.append((title, similarity))  #
Append title and score

    # Sort by similarity in descending order to get the most
similar titles first
    similarity_scores.sort(key=lambda x: x[1], reverse=True)

    # Exclude the original title and get the top N
recommendations
    top_recommendations = [movie[0] for movie in
similarity_scores[1:num_recommendations+1]]
    return top_recommendations

# Example: Get movie recommendations for a specific movie
title
movie_title = "My White Baby"
recommendations = recommend_similar_movies(movie_title)
print(f"Recommended movies for '{movie_title}' based on title
similarity:")
for i, movie in enumerate(recommendations, start=1):
    print(f"{i}. {movie}")
```

Program output:

```
Recommended movies for 'My White Baby' based on title
similarity:
1. Sydney White
2. That's My Boy
3. The Boss Baby
4. The White Ribbon
5. White Fang
```

```
movie_title = "Black Adam"
recommendations = recommend_similar_movies(movie_title)
print(f"Recommended movies for '{movie_title}' based on title
similarity:")
for i, movie in enumerate(recommendations, start=1):
    print(f"{i}. {movie}")
```

Program output:

```
Recommended movies for 'Black Adam' based on title similarity:
1. Black and Blue
2. Patch Adams
3. Ambulance
4. Casablanca
5. Payback
```

4b. Recommendation based on genre

The following section uses fuzzy logic to recommend movies based on user preferences, specifically in a genre-based context. The main idea is to define fuzzy rules that associate a user's genre preference (e.g. "Love", "Neutral", "Hate") with a recommendation score (e.g. "Highly recommended", "Somewhat recommended", "Not recommended"). This method is flexible and can model the vagueness inherent in human preferences.

- Antecedent and Consequent: We define two variables: **User_Preference** (represents how much the user likes or dislikes the genre ranging from -1 to 1, where -1 means "Hate" and 1 means "Love") and **Recommendation** (represents the recommendation score for the movie ranging from 0 to 1, where 0 means "not recommended" and 1 means "highly recommended").
- Fuzzy membership functions: We create fuzzy sets that represent "Hate", "Neutral" and "Love" for user preferences and "Not recommended", "Slightly recommended" and "Highly recommended" for recommendation.
- Fuzzy Rules are defined to match user preferences to recommendation scores: if the user "Loves" the genre, the movie is "Highly Recommended"; if

the user is "neutral" to the genre, the movie is "somewhat recommended"; if the user "hates" the genre, the movie is "not recommended".

- Control system is based on the defined rules as a fuzzy system. The system is used to calculate a recommended score for movies of a certain genre.
- Movie recommendations: For each movie in the specified genre, the system calculates a recommendation score and ranks the movies based on this score to represent the most recommended.

```
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import numpy as np

def create_recommendation_system(df, genre):
    # Create Antecedent and Consequent objects (fuzzy
variables)
    user_preference = ctrl.Antecedent(np.arange(-1, 1.1, 0.1),
'User_Preference')
    recommendation = ctrl.Consequent(np.arange(0, 1.1, 0.1),
'Recommendation')

    # Define fuzzy sets and membership functions for
User_Preference
    user_preference['Hate'] =
fuzz.trimf(user_preference.universe, [-1, -1, 0])  #
Membership for Hate
    user_preference['Neutral'] =
fuzz.trimf(user_preference.universe, [-1, 0, 1])  # Membership
for Neutral
    user_preference['Love'] =
fuzz.trimf(user_preference.universe, [0, 1, 1])  # Membership
for Love

    # Define fuzzy sets and membership functions for
Recommendation
    recommendation['Not Recommended'] =
fuzz.trimf(recommendation.universe, [0, 0, 0.5])  # Not
Recommended
    recommendation['Somewhat Recommended'] =
fuzz.trimf(recommendation.universe, [0, 0.5, 1])  # Somewhat
Recommended
    recommendation['Highly Recommended'] =
fuzz.trimf(recommendation.universe, [0.5, 1, 1])  # Highly
Recommended
```

```
    # Define fuzzy rules based on user preferences
    rule1 = ctrl.Rule(user_preference['Love'],
recommendation['Highly Recommended'])
    rule2 = ctrl.Rule(user_preference['Neutral'],
recommendation['Somewhat Recommended'])
    rule3 = ctrl.Rule(user_preference['Hate'],
recommendation['Not Recommended'])

    # Create the control system with the defined rules
    recommendation_ctrl = ctrl.ControlSystem([rule1, rule2,
rule3])

    # Create a simulation of the control system
    recommendation_sim =
ctrl.ControlSystemSimulation(recommendation_ctrl)

    # Calculate recommendations for each movie in the
specified genre
    recommendations = []

    for index, row in df.iterrows():
        recommendation_sim.input['User_Preference'] =
row[genre]  # Use the genre preference for each movie
        recommendation_sim.compute()  # Compute the
recommendation score
        recommendation_score =
recommendation_sim.output['Recommendation']  # Get the
recommendation score

        # Append the recommendation score to the
recommendations list
        recommendations.append(recommendation_score)

    # Add the recommendations to the DataFrame
    df['pred_Recommendation'] = recommendations

    # Sort movies by recommendation score in descending order
    df = df.sort_values(by='pred_Recommendation',
ascending=False)

    # Display the recommended movies with their scores
    recommended_movies = df[['title', 'Action', 'Adventure',
'Animation', 'Comedy', 'Crime', 'Documentary',
```

```
                              'Drama', 'Family', 'Fantasy',
'History', 'Horror', 'Music', 'Mystery',
                              'Romance', 'Science Fiction', 'TV
Movie', 'Thriller', 'War', 'Western', 'pred_Recommendation']]
    return recommended_movies
```

- set the prefered genre

```
# output for one genre
genre = 'Action'
recommended_movies = create_recommendation_system(data, genre)
print(f"Recommended movies for '{genre}':")
print(recommended_movies)

# output for one genre - for all genres set genres =
data.columns[19:]
genres = data.columns[19:20]  # Exclude the 'MovieTitle'
column
# Generate and print recommendations for each genre, excluding
rows with None values
for genre in genres:
    # Filter out rows with None values in the current genre
column
    filtered_data = data.dropna(subset=[genre])

    if not filtered_data.empty:
        recommended_movies =
create_recommendation_system(filtered_data, genre)
        print('-------------------------')
        print(f"Recommended movies for '{genre}':")
        print(recommended_movies)
        print('-------------------------')
    else:
        print(f"No data for '{genre}', skipping
recommendation.")
```

Program output:

```
Recommended movies for 'Action':
                    title  Action  Adventure  Animation
Comedy  Crime  \
0       Meg 2: The Trench       1          0          0
0      0
4365  In the Line of Fire       1          0          0
0      1
```

```
4372            Safe House        1           0           0
0       0
4373                 Taxi        1           0           0
1       1
4374          King Arthur        1           1           0
0       0
...                   ...      ...         ...         ...
...     ...
6155   A Monster in Paris        0           1           1
1       0
6150        Downton Abbey        0           0           0
0       0
6147               Dracula        0           0           0
0       0
1069      Argentina, 1985        0           0           0
0       0
6612               Brazil        0           0           0
1       0

      Documentary  Drama  Family  Fantasy  ...  Horror  Music
Mystery  \
0                0      0       0        0  ...       1      0
0
4365             0      1       0        0  ...       0      0
1
4372             0      0       0        0  ...       0      0
0
4373             0      0       0        0  ...       0      0
0
4374             0      0       0        0  ...       0      0
0
...            ...    ...     ...      ...  ...     ...    ...
...
6155             0      0       1        1  ...       0      0
0
6150             0      1       0        0  ...       0      0
0
6147             0      1       0        1  ...       1      0
0
1069             0      1       0        0  ...       0      0
0
6612             0      0       0        0  ...       0      0
0
```

```
      Romance  Science Fiction  TV Movie  Thriller  War
Western  \
0           0                1         0         0    0
0
4365        0                0         0         1    0
0
4372        0                0         0         1    0
0
4373        0                0         0         1    0
0
4374        0                0         0         0    1
0
...       ...              ...       ...       ...  ...
...
6155        0                0         0         0    0
0
6150        1                0         0         0    0
0
6147        0                0         0         0    0
0
1069        0                0         0         0    0
0
6612        0                1         0         0    0
0

      pred_Recommendation
0                0.833333
4365             0.833333
4372             0.833333
4373             0.833333
4374             0.833333
...                   ...
6155             0.500000
6150             0.500000
6147             0.500000
1069             0.500000
6612             0.500000

[7651 rows x 21 columns]
------------------------
Recommended movies for 'recommendation_score':
                          title  Action  Adventure  Animation
Comedy  Crime  \
```

```
711                    Inception       1           1           0
0       0
290                Interstellar        0           1           0
0       0
471             The Dark Knight        1           0           0
0       1
119                      Avatar       1           1           0
0       0
494                The Avengers       1           1           0
0       0
...                         ...      ...         ...         ...
...     ...
25212                   The End?       0           0           0
0       0
18116                    Troll 2       0           0           0
0       0
30591  Grande, grosso e Verdone       0           0           0
1       0
25092      Trouble at Timpetill       0           1           0
0       0
38220       The Mother of Tears       0           0           0
0       0

       Documentary  Drama  Family  Fantasy  ...  Horror  Music
Mystery  \
711              0      0       0        0  ...       0      0
0
290              0      1       0        0  ...       0      0
0
471              0      1       0        0  ...       0      0
0
119              0      0       0        1  ...       0      0
0
494              0      0       0        0  ...       0      0
0
...            ...    ...     ...      ...  ...     ...    ...
...
25212            0      0       0        0  ...       1      0
0
18116            0      0       0        1  ...       1      0
0
30591            0      0       0        0  ...       0      0
0
```

```
25092             0       0        1         1  ...         0       0
0
38220             0       0        0         0  ...         1       0
0

       Romance  Science Fiction  TV Movie  Thriller  War
Western  \
711           0                1         0         0    0
0
290           0                1         0         0    0
0
471           0                0         0         1    0
0
119           0                1         0         0    0
0
494           0                1         0         0    0
0
...         ...              ...       ...       ...  ...
...
25212         0                0         0         0    0
0
18116         0                0         0         0    0
0
30591         0                0         0         0    0
0
25092         0                0         0         0    0
0
38220         0                0         0         0    0
0
       pred_Recommendation
711                0.833333
290                0.757300
471                0.723284
119                0.720454
494                0.718439
...                     ...
25212              0.500008
18116              0.500007
30591              0.500007
25092              0.500006
38220              0.500000

[7651 rows x 21 columns]
------------------------
```

4c. Recommendation based on title and genre

The principle behind this code is content-based filtering using cosine similarity to recommend movies based on the similarity of their genre features:

- Feature Matrix contains of rows represent a movie, and the columns represent genre features (e.g., Action, Comedy, Drama, etc.).
- Cosine Similarity is computed between all pairs of movies based on their genre feature vectors. Cosine similarity measures the cosine of the angle between two vectors, and a higher similarity score means the two movies are more alike in terms of their genres.
- Recommendation Function takes a movie title as input and returns a list of recommended movies based on their content (i.e., similarity in genre features). It computes the similarity of the movie with all others and selects the top N most similar movies.

```
from sklearn.metrics.pairwise import cosine_similarity

# Load the dataset
data = data[['title', 'Action', 'Adventure', 'Animation',
'Comedy', 'Crime', 'Documentary',
           'Drama', 'Family', 'Fantasy', 'History', 'Horror',
'Music', 'Mystery',
           'Romance', 'Science Fiction', 'TV Movie',
'Thriller', 'War', 'Western']]

# Prepare the feature matrix (exclude the movie title)
feature_matrix = data.drop(['title'], axis=1)

# Calculate cosine similarity between movies based on genre
features
content_similarity = cosine_similarity(feature_matrix,
feature_matrix)

# Function to get content-based recommendations for a movie
def get_content_recommendations(movie_title,
content_similarity_matrix, num_recommendations=5):
    # Find the index of the movie in the dataset
    movie_index = data[data['title'] == movie_title].index[0]

    # Get the similarity scores for the movie
    movie_similarity_scores =
content_similarity_matrix[movie_index]
```

```
    # Get indices of top N most similar movies (excluding the
movie itself)
    similar_movie_indices =
movie_similarity_scores.argsort()[::-1][1:num_recommendations
+ 1]

    # Get movie titles for the recommendations
    recommendations =
data.iloc[similar_movie_indices]['title'].tolist()

    return recommendations

# Example: Get content-based recommendations for a movie
movie_title = "Meg 2: The Trench"
content_recommendations =
get_content_recommendations(movie_title, content_similarity,
num_recommendations=5)

# Print the content-based recommendations
print(f"Content-Based Recommendations for '{movie_title}':")
for i, recommendation in enumerate(content_recommendations,
start=1):
    print(f"{i}. {recommendation}")
```

Program output:

```
Content-Based Recommendations for 'Meg 2: The Trench':
1. Kill Command
2. Deep Blue Sea 2
3. The Meg
4. Virus
5. Shin Godzilla
```

7.2.2

In the fuzzy logic recommendation system, what does the "User_Preference" variable represent?

- The degree of user's preference for a movie (e.g., Love, Neutral, Hate)
- The average rating of a movie by users
- The genre of the movie
- The release year of the movie

7.2.3

Which of the following statements are true about the fuzzy logic rules in the movie recommendation system?

- If the user has a "Love" preference, the movie will be highly recommended.
- If the user has a "Love" preference, the movie will be rated as "Highly Recommended."
- If the user has a "Neutral" preference, the movie will be rated as "Highly Recommended."
- If the user has a "Hate" preference, the movie will not be recommended.

7.2.4

In the fuzzy logic system, the membership function for the "Love" preference is defined using the fuzzy set **fuzz.trimf**. The "Love" fuzzy set has a membership range from [0, 1, 1]. This means the user will be rated as "Love" when their preference is close to _____, and "Neutral" when it is between _____ and _____.

- 0
- 1
- 0
- 1
- 1
- 0

7.3 Content Based ML Model

7.3.1

Project: Movie recommendation system using machine learning

(by https://medium.com/@saibhargavkarnati/movie-recommendation-system-using-machine-learning-8f6393d71c83)

Dataset

- original: https://github.com/MadScientist29/Movie-Recommendation-System-/blob/main/movies.csv
- local: movies - https://priscilla.fitped.eu/data/recommender_systems/movies_ml.csv

Develop a basic movie recommendation system using Python. By leveraging text data from movie titles and using techniques such as cosine similarity, we created a system capable of generating personalized movie recommendations based on user preferences.

1. Importing Libraries and dataset

```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import warnings
warnings.filterwarnings("ignore")

# loading the data from the csv file to apandas dataframe
movies_data =
pd.read_csv('https://priscilla.fitped.eu/data/recommender_systems/movies_ml.csv')

# printing the first 5 rows of the dataframe
print(movies_data.head())
```

Program output:

```
   index     budget                                         genres
\
0      0  237000000  Action Adventure Fantasy Science Fiction
1      1  300000000                 Adventure Fantasy Action
2      2  245000000                   Action Adventure Crime
3      3  250000000              Action Crime Drama Thriller
4      4  260000000          Action Adventure Science Fiction

                                       homepage      id  \
0                   http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3            http://www.thedarkknightrises.com/   49026
4          http://movies.disney.com/john-carter   49529

                                              keywords
original_language  \
0  culture clash future space war space colony so...
en
```

```
1  ocean drug abuse exotic island east india trad...
en
2         spy based on novel secret agent sequel mi6
en
3  dc comics crime fighter terrorist secret ident...
en
4  based on novel mars medallion space travel pri...
en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                                John Carter

                                          overview
popularity  ... runtime  \
0  In the 22nd century, a paraplegic Marine is di...
150.437577  ...   162.0
1  Captain Barbossa, long believed to be dead, ha...
139.082615  ...   169.0
2  A cryptic message from Bond’s past sends him o...
107.376788  ...   148.0
3  Following the death of District Attorney Harve...
112.312950  ...   165.0
4  John Carter is a war-weary, former military ca...
43.926995  ...   132.0

                                    spoken_languages    status
\
0  [{"iso_639_1": "en", "name": "English"}, {"iso...  Released
1           [{"iso_639_1": "en", "name": "English"}]  Released
2  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},...  Released
3           [{"iso_639_1": "en", "name": "English"}]  Released
4           [{"iso_639_1": "en", "name": "English"}]  Released

                                        tagline  \
0                     Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                          A Plan No One Escapes
3                               The Legend Ends
4            Lost in our world, found in another.
```

```
                                      title vote_average
vote_count  \
0                                    Avatar          7.2
11800
1  Pirates of the Caribbean: At World's End          6.9
4500
2                                   Spectre          6.3
4466
3                    The Dark Knight Rises          7.6
9106
4                              John Carter          6.1
2124

                                                  cast  \
0  Sam Worthington Zoe Saldana Sigourney Weaver S...
1  Johnny Depp Orlando Bloom Keira Knightley Stel...
2  Daniel Craig Christoph Waltz L\u00e9a Seydoux ...
3  Christian Bale Michael Caine Gary Oldman Anne ...
4  Taylor Kitsch Lynn Collins Samantha Morton Wil...

                                                  crew
director
0  [{'name': 'Stephen E. Rivkin', 'gender': 0, 'd...
James Cameron
1  [{'name': 'Dariusz Wolski', 'gender': 2, 'depa...     Gore
Verbinski
2  [{'name': 'Thomas Newman', 'gender': 2, 'depar...
Sam Mendes
3  [{'name': 'Hans Zimmer', 'gender': 2, 'departm...
Christopher Nolan
4  [{'name': 'Andrew Stanton', 'gender': 2, 'depa...
Andrew Stanton

[5 rows x 24 columns]
```

2. Data preprocessing

Before creating our movie recommendation system, we need to prepare the dataset to make it ready for analysis. This includes:

- Exploring the data to understand its structure.
- Selecting important features like titles or genres.
- Filling in any missing values to avoid errors.

- Combining the selected features into one feature vector for similarity calculations.

```
# Number of rows and columns in the DataFrame
print("Number of rows and columns in the DataFrame:",
movies_data.shape)

# selecting the relevant features for recommendation
selected_features =
['genres','keywords','tagline','cast','director']
print("Selected features:", selected_features)

# Handling missing values by replacing them with an empty
string
for feature in selected_features:
    movies_data[feature] = movies_data[feature].fillna('')
```

Program output:

```
Number of rows and columns in the DataFrame: (4803, 24)
Selected features: ['genres', 'keywords', 'tagline', 'cast',
'director']
```

By combining selected features, we create a comprehensive representation of each movie's characteristics, encompassing aspects such as genre, keywords, cast members, and director.

```
# Combining selected features into a single feature vector
combined_features = movies_data['genres'] + ' ' +
movies_data['keywords'] + ' ' + movies_data['tagline'] + ' ' +
movies_data['cast'] + ' ' + movies_data['director']
print(combined_features)
```

Program output:

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                            ...
4798    Action Crime Thriller united states\u2013mexic...
4799    Comedy Romance  A newlywed couple's honeymoon ...
4800    Comedy Drama Romance TV Movie date love at fir...
4801      A New Yorker in Shanghai Daniel Henney Eliza...
```

```
4802    Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

3. Converting text data

In order to effectively use the textual information from the combined elements, we need to convert them into numerical element vectors. This transformation allows us to represent text data in a format suitable for machine learning algorithms.

TfidfVectorizer is often used to convert combined elements into numeric vectors. TfidfVectorizer calculates the TF-IDF (Term Frequency-Inverse Document Frequency) values for each word in the combined features, generating a sparse matrix representing the text data.

By transforming text data into feature vectors, we can effectively capture semantic similarities between movies and facilitate the recommendation process. The resulting feature vectors serve as input to our recommendation system, allowing us to analyze textual content and create personalized movie recommendations for users.

```
# Converting Text Data to Feature Vectors

# Importing the TfidfVectorizer from scikit-learn
from sklearn.feature_extraction.text import TfidfVectorizer

# Creating an instance of TfidfVectorizer
vectorizer = TfidfVectorizer()

# Converting the combined features into feature vectors
feature_vectors = vectorizer.fit_transform(combined_features)

# Printing the feature vectors
print("Feature vectors:", feature_vectors)
```

Program output:

```
Feature vectors:   (0, 2432)    0.17272411194153
  (0, 7755)  0.1128035714854756
  (0, 13024) 0.1942362060108871
  (0, 10229) 0.16058685400095302
  (0, 8756)  0.22709015857011816
  (0, 14608) 0.15150672398763912
  (0, 16668) 0.19843263965100372
  (0, 14064) 0.20596090415084142
  (0, 13319) 0.2177470539412484
```

```
  (0, 17290) 0.20197912553916567
  (0, 17007) 0.23643326319898797
  (0, 13349) 0.15021264094167086
  (0, 11503) 0.27211310056983656
  (0, 11192) 0.09049319826481456
  (0, 16998) 0.1282126322850579
  (0, 15261) 0.07095833561276566
  (0, 4945)  0.24025852494110758
  (0, 14271) 0.21392179219912877
  (0, 3225)  0.24960162956997736
  (0, 16587) 0.12549432354918996
  (0, 14378) 0.33962752210959823
  (0, 5836)  0.1646750903586285
  (0, 3065)  0.22208377802661425
  (0, 3678)  0.21392179219912877
  (0, 5437)  0.1036413987316636
  :    :
  (4801, 17266)     0.2886098184932947
  (4801, 4835)      0.24713765026963996
  (4801, 403)       0.17727585190343226
  (4801, 6935)      0.2886098184932947
  (4801, 11663)     0.21557500762727902
  (4801, 1672)      0.1564793427630879
  (4801, 10929)     0.13504166990041588
  (4801, 7474)      0.11307961713172225
  (4801, 3796)      0.3342808988877418
  (4802, 6996)      0.5700048226105303
  (4802, 5367)      0.22969114490410403
  (4802, 3654)      0.262512960498006
  (4802, 2425)      0.24002350969074696
  (4802, 4608)      0.24002350969074696
  (4802, 6417)      0.21753405888348784
  (4802, 4371)      0.1538239182675544
  (4802, 12989)     0.1696476532191718
  (4802, 1316)      0.1960747079005741
  (4802, 4528)      0.19504460807622875
  (4802, 3436)      0.21753405888348784
  (4802, 6155)      0.18056463596934083
  (4802, 4980)      0.16078053641367315
  (4802, 2129)      0.3099656128577656
  (4802, 4518)      0.16784466610624255
  (4802, 11161)     0.17867407682173203
```

4. Calculating Cosine similarity

To assess the similarity between movies based on their textual content, we use cosine similarity, a common metric to measure the similarity between two vectors. By calculating the cosine similarity, we can determine to what extent each pair of movies is related in terms of their textual properties expressed through feature vectors.

```
# Calculating Cosine Similarity

# Importing the cosine_similarity function from scikit-learn
from sklearn.metrics.pairwise import cosine_similarity

# Getting the similarity scores using cosine similarity
similarity = cosine_similarity(feature_vectors)

# Printing the similarity scores
print("Similarity scores:")
print(similarity)

# Printing the shape of the similarity matrix
print("Shape of the similarity matrix:", similarity.shape)
```

Program output:

```
Similarity scores:
[[1.         0.07219487 0.037733   ... 0.         0.
0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.
0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661
0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.
0.02651502]
 [0.         0.         0.05389661 ... 0.         1.
0.        ]
 [0.         0.         0.         ... 0.02651502 0.
1.        ]]
Shape of the similarity matrix: (4803, 4803)
```

5. Personalization of content

To customize movie recommendations for a user, we need to get the name of their favorite movie and then find the closest match from the movie dataset. This ensures that recommendations are generated based on the user's preferences.

After entering the name of the favorite movie, we create a list containing all the movie names from the dataset and use the function difflib.get_close_matches to find the closest match for the movie name entered by the user. Closest match allows us to ensure that the movie title provided by the user is accurately represented in the dataset. This ensures that we generate recommendations based on the user's intended movie preferences, even if there are minor variations or typos in the input. With the user's favorite movie title identified and the closest match obtained from the dataset, we are ready to proceed with creating personalized movie recommendations based on the selected movie.

```
# getting the movie name from the user
# movie_name = input(' Enter your favourite movie name : ')
# ...or...
movie_name = "Fifth element"

# creating a list with all the movie names given in the
dataset
# Creating a list with all the movie names given in the
dataset
list_of_all_titles = movies_data['title'].tolist()
# print("List of all movie titles:")
# print(list_of_all_titles)

# Finding the close match for the movie name given by the user
find_close_match = difflib.get_close_matches(movie_name,
list_of_all_titles)
print("Close match for the movie name:")
print(find_close_match)

# Extracting the closest match
close_match = find_close_match[0]
print("Closest match:", close_match)
```

Program output:

```
Close match for the movie name:
['The Fifth Element']
Closest match: The Fifth Element
```

6. Find similar movies and generate recommendations

Now that we have identified the user's movie index and obtained the similarity score between all the movies in the dataset, we can proceed to find similar movies and generate personalized recommendations for the user. We'll sort the movies based on their similarity score in descending order, ensuring that the most similar movies appear first in the list. Finally, we list the top recommended movie titles for the user and provide a customized list of movie suggestions based on their preferences.

```
# Finding Similar Movies and Generating Recommendations

# Finding the index of the movie with the closest match title
index_of_the_movie = movies_data[movies_data.title ==
close_match]['index'].values[0]
print("Index of the movie:", index_of_the_movie)

# Getting a list of similar movies based on similarity scores
similarity_score =
list(enumerate(similarity[index_of_the_movie]))
# print("Similarity scores for similar movies:")
# print(similarity_score)

# Sorting the movies based on their similarity score
sorted_similar_movies = sorted(similarity_score, key=lambda x:
x[1], reverse=True)
# print("Sorted similar movies based on similarity score:")
# print(sorted_similar_movies)

# Printing the name of similar movies based on the index
print('Movies suggested for you:')

i = 1
for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = movies_data[movies_data.index ==
index]['title'].values[0]
    if i < 20:
        print(i, '. ', title_from_index, sep='')
        i += 1
    else:
        break
```

Program output:

```
Index of the movie: 322
Movies suggested for you:
1. The Fifth Element
2. The Messenger: The Story of Joan of Arc
3. RoboCop
4. Terminator Genisys
5. Looper
6. Resident Evil: Extinction
7. The Terminator
8. The Helix... Loaded
9. Surrogates
10. Impostor
11. The Adventures of Pluto Nash
12. eXistenZ
13. Cloud Atlas
14. Star Trek
15. Children of Men
16. Ultraviolet
17. Arthur and the Invisibles
18. The 6th Day
19. Daylight
```

7.3.2

Which of the following steps is essential during data preprocessing for a movie recommendation system?

- Filling in missing values to avoid errors.
- Selecting important features like titles or genres.
- Performing clustering of users.
- Generating predictions for user ratings.

7.3.3

Why is it necessary to convert text data into numerical vectors for a movie recommendation system?

- To enable the use of machine learning algorithms.
- To capture semantic similarities between movies.
- To speed up the rendering of movie posters.
- To ensure compatibility with csv files.

7.3.4

What does the cosine similarity metric measure in the recommendation system?

- The similarity between feature vectors of two movies.
- The number of genres shared by two movies.
- The popularity difference between two movies.
- The difference in release years of two movies.

7.3.5

What is the role of **difflib.get_close_matches** in personalizing recommendations?

- To find the closest match for the user's input movie title.
- To handle minor variations or typos in the user's input.
- To sort the movies alphabetically.
- To calculate the cosine similarity score.

7.3.6

What happens after calculating similarity scores in a movie recommendation system?

- Movies are sorted by similarity in descending order.
- A customized list of top similar movies is created.
- The dataset is split into training and testing sets.
- User reviews are added to the dataset.

7.3.7

Project: Second Movie recommendation system using machine learning

Develop a machine learning recommendation system

Dataset

- original: https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies
- reduced local: https://priscilla.fitped.eu/data/recommender_systems/movies_dataset.csv

Create a functional movie recommendation system based on machine learning in which the knowledge and techniques presented so far will be used and applied to the provided data set.

Start by carefully analyzing your dataset to understand its structure, the types of data it contains, and how it can be used to generate meaningful recommendations. This step is critical to identifying the features that will be most relevant to your system, such as movie titles, genres, or other metadata.

Once you have a thorough understanding of the data set, proceed with pre-processing the data. Resolve any missing values to ensure that the data set is complete and suitable for analysis. You will also need to transform the data into a format that machine learning algorithms can work with. Combine selected features into a unified representation that captures the key characteristics of each film.

After preprocessing, choose the appropriate approach to create a recommendation system. You can use content-based filtering, collaborative filtering, or a hybrid approach, depending on what best suits your dataset and project goals. Implement a method to calculate similarity, which allows to compare movies and determine which are most similar.

The final step is to design a way to display personalized recommendations. This includes allowing the user to enter their favorite movie, find the closest match within a dataset, and generate a ranked list of recommended movies based on similarity.

The project goal is to provide a complete and functional recommendation system that demonstrates your ability to analyze data, effectively pre-process it, apply machine learning techniques and present the results in a user-friendly way.

1. Import libraries and dataset

```
# imports
```

2. Data understanding and preprocessing

```
# data inspection and cleaning
```

3. Feature extraction

```
# feature extraction
```

4. Conversion and similarity method application

```
# conversion and ML method application
```

5. User interface and recommendation

```
# final step
```

Multi-Criteria Decision Analysis

Chapter 8

8.1 MCDA

📖 8.1.1

Multi-criteria decision analysis

Multi-Criteria Decision Analysis (MCDA) is a structured approach to decision-making that helps identify the best option among a range of alternatives by evaluating multiple criteria. Unlike single-criterion methods, which consider only one factor, MCDA takes into account various criteria that may each hold different levels of importance. This method is commonly used in recommender systems or decision support systems, where making the right choice requires analyzing complex and diverse factors. The core goal is to select the most suitable alternative, considering both quantitative and qualitative aspects that contribute to an informed decision.

An important aspect of MCDA is that it treats each criterion individually rather than combining all criteria into a single, aggregate score. Each criterion can have a unique scale and unit, and they can be maximized or minimized depending on the context. For example, while evaluating a family car, criteria like fuel efficiency (to be maximized) and cost (to be minimized) cannot be measured on the same scale. Therefore, each must be evaluated separately to determine its contribution to the decision.

Expert knowledge often plays a key role in the MCDA process, helping ensure that each criterion is appropriately considered and weighed. Some common applications of MCDA include selecting the best software tool for a project, choosing an optimal family car, or deciding on a location for a new factory. Each scenario requires analyzing a unique set of criteria, and MCDA allows decision-makers to systematically assess options to arrive at the most suitable choice.

📝 8.1.2

Which of the following describes a primary characteristic of MCDA?

- It evaluates alternatives based on multiple criteria.
- It aggregates all criteria into one score.
- It only considers cost as the deciding factor.
- It ignores expert knowledge.

8.1.3

MCDA is useful in decision-making because it evaluates options based on _____ criteria, helping to select the most _____ alternative in situations where criteria _____ be aggregated into a single score.

- suitable
- cannot
- multiple
- can
- simple

8.1.4

MCDA involves several key steps to ensure a thorough evaluation of all alternatives.

1. The first step is **structuring the decision problem**. This means defining the objectives, understanding the decision context, and identifying the alternatives available for consideration.
2. Once the problem is structured, the second step is to **specify the criteria** that will be used to evaluate each option. Criteria selection is essential as it directly influences the effectiveness of the final decision.
3. After establishing the criteria, the next step is to **measure each alternative's performance** against these criteria. This often involves gathering data or scoring the alternatives based on expert judgment.
4. Once scores are assigned, the fourth step is to **apply weights to the criteria**. Weighting reflects the relative importance of each criterion - some may be critical to the decision, while others are less significant.
5. In the final step, decision-makers **apply the scores and weights to rank the alternatives** and arrive at a supported decision.

By following these steps, MCDA provides a structured approach to decision-making that can handle complex situations involving multiple, often conflicting criteria. It enables decision-makers to systematically analyze each factor, promoting transparency and consistency in the decision-making process.

8.1.5

Which steps are involved in MCDA?

- Structuring the decision problem
- Scoring alternatives and weighting criteria

- Ignoring alternative options
- Aggregating criteria into one metric

8.1.6

Typical order of main steps of decision-making analysis is:

- Measuring alternatives' performance
- Applying scores and weights to rank alternatives
- Structuring the decision problem
- Specifying criteria
- Scoring alternatives and weighting the criteria
- Supporting decision-making

8.1.7

In MCDA, after structuring the decision problem, the next step is to _____ criteria, followed by measuring each alternative's _____, and finally _____ the scores and weights.

- performance
- ranking
- specify

8.2 Initial steps

8.2.1

The first step in MCDA involves determining all possible alternatives, also called "variants," for a solution. This step is foundational, as it defines the range of choices that decision-makers can evaluate. To start, it's necessary to gather every option that meets the basic requirements of the decision context. For instance, if we are choosing a new software system, the first task is to list all available software options that meet our technical and functional needs. This preliminary filtering ensures that only viable candidates move to the next stage.

In practical terms, determining alternatives requires two key actions: applying specific selection procedures and conducting a thorough search to capture all options. Applications for selection procedures might include setting minimum qualification standards or technical requirements that each alternative must meet. Additionally, the search for alternatives should be exhaustive - this could mean

consulting industry sources, seeking expert recommendations, or reviewing product databases to ensure all relevant options are considered.

Expert knowledge is also valuable during this stage, as experts may provide insights into alternatives that might not be immediately apparent. Including expert input helps avoid overlooking options that may be beneficial but less obvious. By considering these factors, MCDA establishes a broad foundation for a well-informed decision-making process.

8.2.2

Which of the following is **NOT** typically a criterion in MCDA?

- Personal preference of the decision-maker
- Cost
- Performance measurement
- User-friendliness

8.2.3

What are some possible criteria in MCDA?

- Frequency of support inquiries
- Arbitrary personal bias
- Compatibility
- Random selection of options

8.2.4

The second step in MCDA is to set criteria for evaluating each alternative. Criteria are crucial, as they provide the standards by which each option is assessed. In MCDA, we prefer using aggregate criteria that encapsulate broader aspects of the decision, which allows for a holistic evaluation. For instance, when evaluating potential job candidates, aggregate criteria might include adaptability, experience, and skill level. Criteria should be carefully chosen to ensure that all significant aspects of the decision are addressed, thus creating a balanced evaluation framework.

These criteria must be comprehensive, covering all relevant dimensions of the problem. If the decision involves purchasing equipment, criteria might include cost, durability, efficiency, and ease of maintenance. This ensures that the decision-making process takes into account every important factor. Additionally, criteria must be independent, meaning each one addresses a unique aspect of the problem

without overlap, allowing each option to be evaluated on its own merits across all dimensions.

Expert knowledge can be valuable in defining these criteria. Experts can help identify essential factors and refine the criteria to better reflect the needs of the situation. By establishing clear, relevant, and independent criteria, decision-makers can ensure that each alternative is assessed fairly and comprehensively.

8.2.5

Which type of criteria is preferred in MCDA for evaluation?

- Aggregate criteria
- Minimizing only cost
- Single-factor criteria
- Ignoring all primary criteria

8.2.6

What is a requirement for criteria to ensure a balanced assessment?

- Criteria must be independent
- Criteria must all focus on cost
- Criteria should overlap
- Criteria should exclude expert input

8.3 Weighing criteria

8.3.1

The first critical step in Multi-Criteria Decision Analysis (MCDA) is forming an expert group to ensure a well-rounded perspective on the problem at hand. This group should represent all significant user groups who will be affected by the decision. It's essential to achieve a balanced representation across user groups, as having too many members from one group could bias the assessment. By selecting diverse experts, the decision-making process can reflect a broad range of viewpoints, leading to more equitable and comprehensive criteria evaluation.

After forming the expert group, the next step is to determine the significance or weight of each criterion in the decision process. Multiple methods are available for assessing criterion significance, including:

- Ordering criteria
- Sorting criteria
- Fuller's triangle
- Saaty's method

Each of these methods has specific advantages depending on the complexity and requirements of the decision. For example, ordering and sorting are relatively straightforward methods, while Fuller's triangle and Saaty's method provide a more structured approach that can capture nuanced preferences among criteria.

Once the method is chosen, each expert is asked to give their input on the relative importance of each criterion. This data is then analyzed to calculate an overall weight for each criterion, reflecting its importance in the decision-making process. This systematic approach helps ensure that the final ranking of alternatives considers the perspectives of all key stakeholders. Assigning proper weights to criteria is essential to create a balanced and objective decision-making framework.

8.3.2

Which of the following methods are commonly used to determine the significance of criteria in MCDA?

- Ordering criteria
- Saaty's method
- Using cost as a factor
- Ignoring expert opinions

8.3.3

Ordering criteria

Ordering criteria is a very easy method. Each expert determines a clear order of importance of the criteria.

This evaluation will be stored in the table:

expert	criteria			
		1	...	j
	1			
	...			
	i			α_{ij}
	$\sum_i \alpha_{ij}$			

The significance of every criterion will be calculated:

$$B_j = 1 - \frac{\sum_i \alpha_{ij}}{\sum_j \sum_i \alpha_{ij}}$$

This method is very easy, but has some disadvantages:

- It is too strict, an expert cannot evaluate two criteria in the same order
- Distance between criteria is constant, expert cannot evaluate some criterion as much more important than any other criterion.

📖 8.3.4

Scoring criteria

Experts will evaluate every criterion by a point value from a given range with the possibility of repetition.

expert	criteria				
		1	...	j	$\sum_j \beta_{ij}$
	1				
	...				
	i			β_{ij}	

This evaluation could be converted to weights for individual experts:

$$p_{ij} = \frac{\beta_{ij}}{\sum_j \beta_{ij}}$$

Next could be determined the significance of every criterion:

$$B_j = \sum_i p_{ij}$$

Some questions need to be answered. First is the problem of the appropriate choice of point range:

- we need a large enough range (twice the number of criteria could be appropriate),
- too large scale will be a problem for experts, they can only use a part of it (for example 100 points for 3 criteria),
- mostly "natural" ranges (1 ÷ 5, 1 ÷ 10, ...) are suitable.

The second question is about incorporating a null value. Usually, it is not included. Every criterion has some importance.

The danger of using part of the scale range is eliminated by conversion to weights.

🕮 8.3.5

Fuller's triangle

Fuller's triangle (**Pairwise comparison method**) is very easy for experts to use:

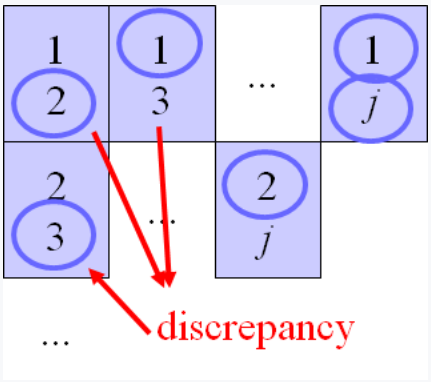
- we will compile all pairs of criteria,
- the expert determines the more important of the pair only,
- it is possible to leave the pair equally significant.

Evaluation by an expert:

- more significant of the pair = 1 point,
- both equally significant = 0.5 points.

A typical application of Fuller's triangle - each expert receives a list of all pairs of criteria and circles the more significant one in each pair. If he considers both criteria equally important, he circles both.

Unfortunately, the expert can commit an inconsistency in the assessment, as shown in the figure:



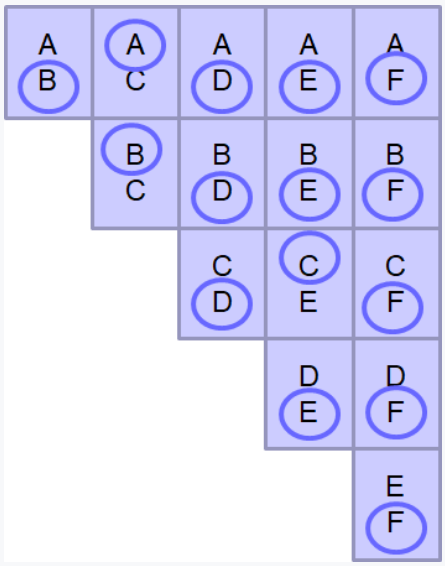
If we want to detect this discrepancy, we can use a simple application from Graph Theory that was presented in [Farana 2016]. We will use a graph (complete graph) to describe the criteria:

- Vertices represent criteria.
- Edges represent individual ratings.
- We orient the edges to a more significant criterion.

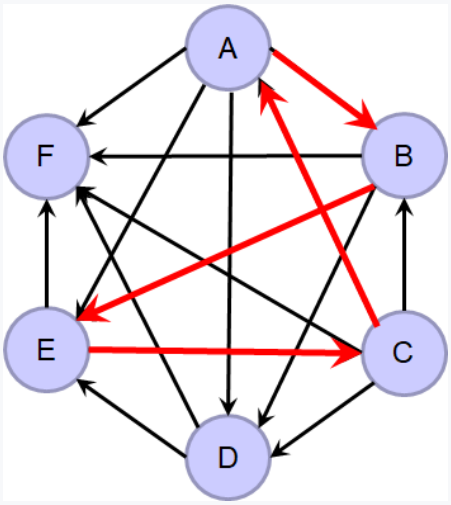
If the resulting graph is acyclic, no conflict has occurred.

We will apply acyclicity checking by numbering the vertices from start (with zero input degree) to the end (with zero output degree) vertex.

An example of conflicts in expert evaluation:



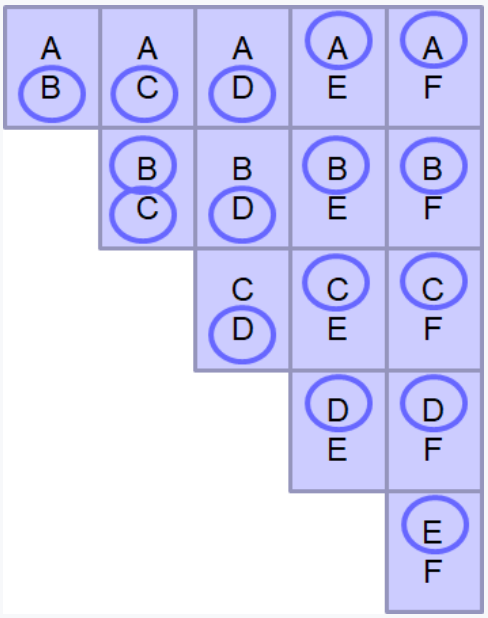
The resulting graph, constructed according to the description above:



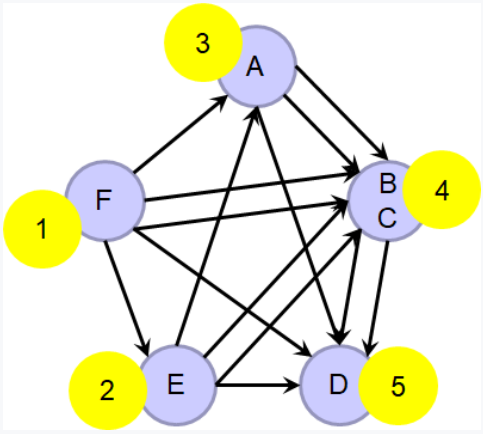
Vertices cannot be numbered, the graph is cyclic.

The only thing we can do is discard this expert's rating as unusable.

The next example presents what we will do when two criteria are evaluated by the same significance.



When building the graph, we will merge vertices with the same rating: Parallel edges must be oriented in the same way (they must be multiple):



It is evident, that in this example we can enumerate the vertices, and this criteria evaluation is correct.

The number of points for every criterion will be filled into the table:

	criteria			
expert		1	...	j
	1			
	...			
	i			γ_{ij}
	$\sum_i \gamma_{ij}$			

And the significance of every criterion will be calculated:

$$B_j = \frac{\sum_i \gamma_{ij}}{i}$$

📖 8.3.6

Saaty's method

Saaty's method (Quantitative pairwise comparison) is also based on the evaluation of pairs of criteria. All pairs of criteria are compared and the evaluation is stored in the Saaty matrix **S** = (s_{ij}) according to the following system:

$$(s_{ij}) = \begin{cases} 1 & \text{- } i \text{ and } j \text{ are equivalent} \\ 3 & \text{- } i \text{ is weakly preferred over } j \\ 5 & \text{- } i \text{ is strongly preferred over } j \\ 7 & \text{- } i \text{ is very strongly preferred over } j \\ 9 & \text{- } i \text{ is absolutely preferred over } j \end{cases}$$

Values 2, 4, 6, and 8 are left for intermediate grades.

It is obvious that *sii* = 1 since the criterion is equal to itself.

It must hold that $s_{ji} = 1/s_{ij}$ for all *i, j*.

The value *sij* represents the approximate ratio of the weights of criteria *i* and *j*, in mathematical notation:

$$s_{ij} \approx \frac{v_i}{v_j}$$

Based on this evaluation we will use Saaty's method procedure:

First, we fill in the Saaty matrix:

1. There will be ones on the diagonal ($s_{ii} = 1$).
2. $s_{ij} \in \langle 1, 9 \rangle$, if *i* preferred over *j*.
3. $s_{ji} = \frac{1}{s_{ij}}$

We calculate the value s_i for each *i:*

$$s_i = \prod_{j=1}^{k} s_{ij}$$

We calculate for each *i:*

$$R_i = (s_i)^{1/k} = \sqrt[k]{s_i}$$

Next, we calculate:

$$\sum_{i=1}^{k} R_i$$

Finally, we determine the criteria weights by the expression:

$$v_i = \frac{R_i}{\sum_{i=1}^{k} R_i}$$

Because of the overdetermination of the system, the matrix S must be "satisfactorily consistent", with the variance estimation:

$$\sigma^2 = \frac{F}{d} = \frac{\sum_{i=1}^{k}\sum_{j>i}\left(\ln s_{ij} - \left(\ln v_i - \ln v_j\right)\right)^2}{\frac{(k-1)(k-2)}{2}}$$

$$\sigma^2 < 0{,}1 \text{ for } k = 3$$

$$\sigma^2 < 0{,}2 \text{ for } k = 4, 5, 6, 7$$

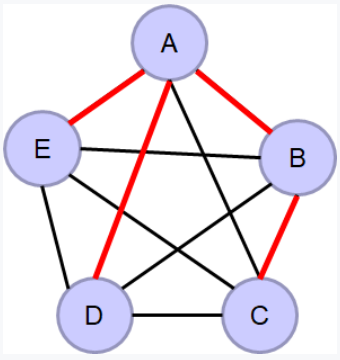
$$\sigma^2 < 0{,}3 \text{ for } k > 7$$

If we want to eliminate the overdetermination, we can use another method from Graph Theory.

It is based on the answer to a significant question: What is the minimum number of evaluations of pairs of criteria and which pairs should be compared?

The total number of comparisons represents a complete graph having $\frac{k(k-1)}{2}$ edges.

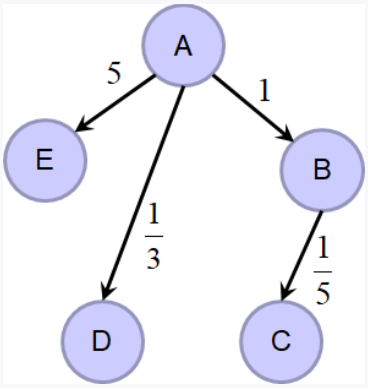
The minimum number of pairs than its spanning tree (any) having k – 1 edges:



We get an evaluation of the selected pairs, e.g.:

- A – B = 1
- B – C = 1/5
- A – D = 1/3
- A – E = 5

We will include the corresponding edges in the graph and mark their orientation:

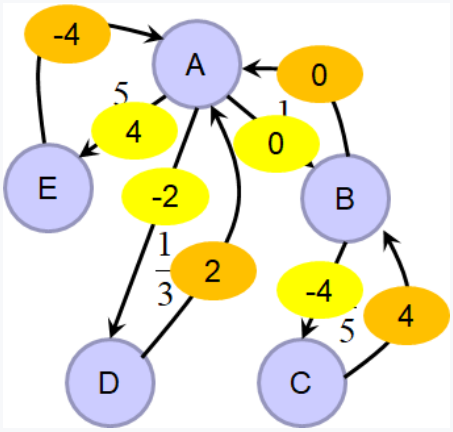


We will replace the evaluation with points according to the table:

weight	points
$\frac{1}{9}$	-8
$\frac{1}{8}$	-7
...	...
$\frac{1}{2}$	-1
1	0

2	1
3	2
...	...
8	7
9	8

Next, we will add oppositely oriented edges with opposite evaluations:



We fill in Saaty's table:

We add up the points on the way from X to Y and convert them into a rating using the same table.

We will designate the lower part of the table as the evaluation of inverted values.

	A	B	C	D	E
A	1	1	1/5	1/3	5
B	1	1	1/5	1/3	5
C	5	5	1	3	9
D	3	3	1/3	1	7
E	1/5	1/5	1/9	1/7	1

Next, we proceed as standard:

Saaty matrix								weight
	A	B	C	D	E	s_i	R_i	v_i
A	1	1	0,2	0,333333	5	0,333333	0,8027	0,10917
B	1	1	0,2	0,333333	5	0,333333	0,8027	0,10917
C	5	5	1	3	9	675	3,6801	0,50046
D	3	3	0,333333	1	7	21	1,8384	0,25001
E	0,2	0,2	0,111111	0,142857	1	0,000635	0,2294	0,03119

Note: Using this procedure, we can get a rating higher than 9, or lower than 1/9.

8.3.7

The most sofisticated method for evaluation the criteria importance is:

- Saaty's method.
- Satyr's method.
- Sander's method.

8.3.8

What is the primary purpose of weighting criteria in MCDA?

- To prioritize key factors in the decision
- To eliminate less important alternatives
- To increase the number of criteria used
- To randomly assign importance

8.4 Evaluating the alternatives

📖 8.4.1

In MCDA, after setting criteria, the next essential step is to evaluate the alternatives according to these criteria. This evaluation step ensures that each alternative is analyzed based on how well it meets each criterion, providing a clear comparison between options. Achieving objective evaluations, however, can sometimes be challenging. When objective data isn't readily available, expert knowledge becomes crucial in filling information gaps. Experts' insights allow us to assess qualitative aspects and make informed estimates, particularly in complex decision scenarios.

To ensure accuracy, verification from multiple independent sources is also beneficial. By consulting different sources, we gain a more comprehensive understanding, which reduces potential biases and enhances the reliability of the evaluations. Independent verification is especially important when relying on subjective judgments or when data sources may vary in reliability. This cross-validation approach strengthens the foundation of our decision-making process.

For MCDA to be effective, each evaluation must reflect the criteria defined earlier. This consistency ensures that all significant aspects of each alternative are considered fairly, contributing to an objective and balanced assessment. Evaluating alternatives in this way allows decision-makers to weigh options against predefined benchmarks, ultimately leading to more robust, data-supported conclusions.

📝 8.4.2

Which approach is used to enhance the objectivity of evaluating alternatives?

- Consulting independent sources
- Ignoring qualitative factors
- Relying solely on data from one source
- Using only cost-based criteria

📝 8.4.3

The _____ of multiple independent sources helps to achieve objective values when evaluating alternatives.

- verification
- reduction
- selection
- elimination

🕮 8.4.4

The final step in MCDA is ranking the alternatives based on how well they meet the established criteria. This ranking allows decision-makers to see which options are most suitable according to the defined criteria and weights. Ranking methods are essential to translate the evaluations into a clear preference order, enabling a straightforward comparison between alternatives.

Two popular methods for ranking alternatives in MCDA are:

- The partial order method ranks options by establishing relationships among them without forcing them into a strict hierarchy. This approach is helpful when alternatives are comparable but not necessarily in a linear order. It provides flexibility, allowing decision-makers to see clusters or groups of similar choices rather than a single best option.
- The method of the base, on the other hand, involves selecting a "base" alternative that serves as a reference point. Each alternative is then compared to this base, allowing for a more structured ranking system. This method is straightforward and effective, especially when one alternative is clearly superior or a benchmark needs to be maintained.

🕮 8.4.5

Partial order method

The partial order method (weighted by criteria importance) is the easiest method for ranking the alternatives.

According to the evaluation, we will determine the order of the alternatives for every criterion.

		criteria			
		1	...	j	
alternatives		B_1		B_j	P_i
	1				
	...				
	i			h_{ij}	

According to the filled table, we determine the weighted ranking of every alternative:

$$P_i = \sum_j B_j h_{ij}$$

The last step is to determine the order of the alternatives. Obviously, the lower the weighted rank, the better.

🕮 8.4.6

Method of base

For each criterion we determine the base value (minimum, maximum, 0, most often the average):

$$h_{Bj} = \frac{\sum_i h_{ij}}{n}.$$

		criteria		
		B_1		B_j
		1	...	j
alternatives	1			
	...			
	i			h_{ij}
	h_{Bj}			

Next, we recalculate the values, based on the type of criterion:

- cost-based criterion (-):

$$z_{ij} = \frac{h_{Bj}}{h_{ij}} B_j$$

- beneficial-based criterion (+):

$$z_{ij} = \frac{h_{ij}}{h_{Bj}} B_j$$

Next, we can calculate the usefulness of all alternatives:

$$S_i = \sum_j z_{ij}$$

Obviously, the higher value the alternative achieves, the better.

📖 8.4.7

Some special ideas

A typical question is if it has sense to use more different methods for ranking alternatives and comparing the obtained results.

The answer is obvious. No. Each method has a different basis, and results may be different.

Some problems are very sensitive to changing the values of parameters or the significance of criteria. The problem of the sensitivity of the task to the value of the coefficients of significance can be evaluated:

- try small value changes (± 10 %).

📝 8.4.8

When using the method of the base, the best alternative will have the usefulness of all alternatives

- highest
- lowest
- average value

📝 8.4.9

Which ranking method in MCDA uses a reference point or benchmark for comparison?

- Method of the base
- Partial order method
- Elimination method
- Random selection method

📝 8.4.10

In MCDA, the _____ method establishes relationships among alternatives without forcing them into a strict hierarchy.

- partial order
- base

- elimination
- cost

8.5 Graph theory methods applicability in MCDA

📖 8.5.1

1. Introduction

Methods of multi-criteria analysis of variants (multi-criteria decision-making) are described above.

We use the knowledge of experts to choose the best option while respecting a large number of, often conflicting, criteria. Above all, in the phase of determining the importance of individual criteria. Several methods have been gradually developed that try to help experts in determining the importance of criteria, especially when there are more of them. The Fuller’s triangle and the Saaty's method are particularly effective methods. Unfortunately, the facilitation of expert decision-making is balanced by the danger of conflicting evaluation, which can negatively affect the result of the entire multi-criteria analysis, because we know from practice that these tasks are often very sensitive to the significance of the criteria. In other words, even a small change in the significance of the criteria can cause a significant change in the resulting evaluation of the variants.

Previous chapters presented how we can effectively use methods known from Graph Theory, to detect these inconsistencies or even avoid them. This study presents practical experiences from the use of these methods.

📖 8.5.2

2. Fuller’s triangle

Fuller's triangle (also called the Pairwise comparison method), is a way to compare and determine the significance of a large number of evaluation criteria. The expert is presented with a set of all pairs of criteria with a request to mark which of the pair is more important, or may also mark both as equally important. This greatly simplifies his decision-making on the one hand, but at the same time, there is a danger that his opinion will be inconsistent.

This contradiction cannot be resolved and in more complex cases it is possible that it will not even be detected. For this, we can advantageously use a graph (complete), in which the vertices represent the criteria and the edges of their mutual evaluation. Firstly, we use the method of gluing for vertices with the same importance, next we

orient the edges towards a more significant criterion. The resulting graph must be acyclic. If the graph contains cycles, the evaluation is inconsistent.

Table 1: Results of using Fuller's triangle in implemented multi-criteria decision-making tasks

Number of criteria	Number of experts	Expert experience	Number of problems	Longer cycle length
7	7	Experienced	0	0
9	8	Very experienced	0	0
8	6	Inexperienced	1	3
11	10	Experienced	1	3
7	8	Inexperienced	0	0
9	7	Inexperienced	0	0
8	5	Experienced	0	0
8	7	Very experienced	0	0
9	7	Inexperienced	1	3
10	12	Experienced	1	4
12	11	Experienced	0	0
7	7	Very experienced	0	0
8	7	Very experienced	0	0
7	9	Inexperienced	1	3
9	7	Experienced	0	0
9	9	Experienced	0	0
10	8	Inexperienced	0	0
8	6	Very experienced	0	0
7	7	Experienced	0	0
8	5	Experienced	0	0
9	7	Very experienced	0	0
12	8	Experienced	2	3
8	7	Experienced	0	0
7	5	Very experienced	0	0
9	7	Inexperienced	0	0

Table 1 presents the obtained results of using Fuller's triangle in implemented multi-criteria decision-making tasks with 7 or more criteria. It is evident that a very small number of problems have been identified. The typical length of the cycle is 3. Experts with little experience made the most mistakes.

A structured interview method was used to identify the source of faults. Most often, the experts stated that they found the criteria to be very similarly important, but for some reason, they did not mark them as equally important. This also shows the possibility of removing the cycle by marking all criteria in the cycle as equally important.

An interesting fact was discovered during the discussion with very experienced experts. A number of them stated that they were aware of the danger of inconsistent evaluation. And they prevent it by ranking the criteria in order of importance before

they begin to fill in Fuller's triangle. Then the question is whether to use directly the method of order to determine the significance of the criteria.

🕮 8.5.3

3. Saaty's method

Saaty's method is a well-known method that enables a more sensitive evaluation of criteria. Expert is evaluating every pair of criteria using values from 1 to 9 to determine the strength of preferences. Since all pairs of criteria are compared, the same problem as in Fuller's triangle can arise, i.e. the creation of a cycle. Then also the same method to identify the cycle, described above, can be used. However, the possibility to express the importance of preferences brings additional risks of inconsistency in evaluation. To eliminate overdetermination of evaluations, it is possible to use a graphic interpretation of the relationships between the criteria in the form of a graph. In this graph, we will find the spanning tree (any of them), we will evaluate criteria pairs at the spanning tree only, and calculate the rest of the evaluations.

Table 2 presents an example of five criteria (A to E) evaluation when only pairs A-B, B-C, A-D, and A-E were given by an expert and all rest values were calculated. It is evident, that the main diagonal contains values of 1, and the reciprocal values we obtain according to the relation (B-A) = 1/(A-B).

Table 2: Saaty’s table calculated from the expert’s evaluations

	A	B	C	D	E
A	1	1	1/5	1/3	5
B	1	1	1/5	1/3	5
C	5	5	1	3	9
D	3	3	1/3	1	7
E	1/5	1/5	1/9	1/7	1

The next procedure already respects Saaty’s method, so we determine the weights of the criteria and we can check the consistency of Saaty’s matrix by calculating the variance estimate. The resulting value for the example from Table 2 is 0.155, and for five criteria a variance estimate of less than 0.2 is required.

Fifteen decision tasks with seven or more criteria were analyzed and experts were asked about their opinion on this method. The obtained results were significantly dependent on the experience of the experts:

- Very experienced - neutral opinion, they do not see a problem in the over determination of the assessment, and they are not concerned about inconsistent assessment.
- Experienced - predominantly positive opinion, they see the danger in the over determination of the assessment, and they are concerned about inconsistent assessment.
- Inexperienced - a completely positive opinion, they see a danger in the over determination of the assessment, and they are very concerned about inconsistent assessment. They prefer to provide as few ratings as possible.

🕮 8.5.4

4. Results and Conclusion

The study presented two possible applications of algorithms from graph theory using classical methods for determining the significance of criteria. Sophisticated methods of evaluating the importance of criteria such as Fuller's triangle or Saaty's method carry the risk of inconsistent evaluation. The presented applications of algorithms from graph theory allow inconsistencies to be detected or even eliminated.

The use of graph algorithms gives new options to support decision-making, many publications are focused on this area. But classical methods are still used and the use of presented procedures and applications from graph theory can significantly support them. The results achieved and the opinions of experts, presented in this post, prove it. This concept was presented in [Farana 2024].

Projects

Chapter 9

9.1 Projects

🕮 9.1.1

Project: Web portal development

The company is planning to install a new web portal and has announced a tender. Four companies have applied, and you need to select the most suitable candidate for this project. The decision will be based on evaluating the companies using four criteria:

1. Price in CZK
2. Realization time in months
3. References (number of previous successfully realized projects)
4. Quality of the design (percentage of fulfillment of requirements)

The goal of this assignment is to use Multi-Criteria Decision Analysis (MCDA) to evaluate and rank these four companies based on the given criteria. You will need to normalize the data, assign weights to the criteria, score each alternative, and provide a final ranking.

Following table presents input data.

	Criteria			
Alternative	Price [CZK]	Realization time [month]	References [number]	Quality [points]
1	80000	12	0	70
2	160000	12	9	80
3	180000	15	5	65
4	240000	7	12	95

1. Normalize the criteria

Normalization is not strictly necessary in every situation, but it is generally **recommended** and often considered an essential step for the following reasons:

- Consistency of Units: Different criteria may have different units of measurement (e.g., price in CZK, time in months, design quality in percentage points). Without normalization, these criteria can't be directly compared or combined because they are on different scales. Example: Price may range from 10,000 CZK to 50,000 CZK, while design quality is scored

between 0-100%. Normalizing these scales brings them to a common framework, allowing for accurate comparison.

- Fair Comparison: Criteria like cost and realization time are typically minimized, meaning lower values are better. In contrast, design quality and references are typically maximized, meaning higher values are better. Normalization ensures that these opposing criteria are treated in a consistent manner, making it clear whether an alternative is performing better or wrse across all the criteria.
- Combining Diverse Criteria: MCDA often involves combining multiple criteria with different scales (e.g., cost in money, time in months, quality in percentage). Without normalization, the decision-making process could disproportionately emphasize certain criteria, leading to an inaccurate final decision.

For a simpler solution process, we can skip normalization in this task.

2. Assign weights to the criteria

To obtain the weighting of the criteria we will organize a group of four experts and ask them to order the criteria from most important (1) to least important (4)

Their opinions are fulfilled to the table, to verify the fulfillment correctness, the checksum in every row must be 10:

	Criteria				
Expert	Price	Realization time	References	Quality	Checksum
E1	2	3	4	1	10
E2	4	3	2	1	10
E3	1	3	4	2	10
E4	1	4	3	2	10
Sum	8	13	13	6	40
Bj	0,8	0,675	0,675	0,85	

To obtain the significance of every criterion we count the sum in every column and count B_j = 1- (sum in column/sum of all columns)

3. Score the alternatives

To evaluate all alternatives we will use the partial ordering method. So we will determine the type of criterion and order all alternatives from 1 to 4:

	Criteria			
Alternative	Price	Realization time	References	Quality
1	1	2,5	4	3
2	2	2,5	2	2
3	3	4	3	4
4	4	1	1	1
Bj	0,8	0,675	0,675	0,85

4. Rank the alternatives

According to the Partial ordering method we will now multiply all values by criteria importance B_j and cout the sum in every row P_i:

	Criteria					
Alternative	Price	Realization time	References	Quality	Pi	Order
1	0,8	1,6875	2,7	2,55	7,7375	3
2	1,6	1,6875	1,35	1,7	6,3375	2
3	2,4	2,7	2,025	3,4	10,525	4
4	3,2	0,675	0,675	0,85	5,4	1

The lowest value indicates the best alternative.

5. Provide a justification for your decision

After ranking the alternatives, **justify your decision**. Explain why the selected company was chosen, highlighting the most significant factors that led to this conclusion.

Consider providing a brief analysis of the trade-offs between the different criteria and how they influenced your final ranking.

🕮 9.1.2

Project: Place for nuclear power station

The energy sector plays a pivotal role in meeting the growing demands for electricity while ensuring environmental sustainability. As part of the national strategy for enhancing energy security and reducing reliance on fossil fuels, the government has identified the need for expanding nuclear power generation. To achieve this goal, a

comprehensive evaluation of potential sites for new nuclear power stations is required.

This project involves the evaluation of three potential sites for the construction of new nuclear power stations. The sites were selected based on their geographical and logistical suitability, but further detailed analysis is required to determine which site offers the best long-term value and meets national energy policy goals. The evaluation process will consider several critical factors, including economic costs, operational efficiency, and social impact.

To make an informed decision, three possible locations have been evaluated against six key criteria. These criteria will serve as the foundation for comparing the alternatives and selecting the most appropriate site for development. The following criteria have been identified for the evaluation:

- **Number of Workers:** This criterion measures the number of employees required for the operation of the nuclear plant. A larger workforce might indicate a more complex facility, while a smaller workforce could point to automation and efficiency but may also affect local employment rates.
- **Maximal Power in MW:** This refers to the maximum electricity output the plant can generate in megawatts. A higher power output increases the contribution of the plant to the national grid, making it a more critical asset for energy security.
- **Investment in Billions CZK (mld. Kč):** This criterion accounts for the initial investment required for the construction of the nuclear plant. It includes costs related to land acquisition, construction, installation, and safety measures. The lower the investment, the quicker the return on investment may be realized, but it might come at the expense of certain design features or capacity.
- **Yearly Operating Costs in Millions CZK (mil. Kč):** This criterion reflects the annual operating costs necessary to keep the nuclear plant running. This includes maintenance, staff wages, fuel costs, and other operational expenses. Lower operating costs generally indicate higher operational efficiency and better long-term financial sustainability.
- **Number of Evacuated Residents:** This represents the number of people who would need to be evacuated in the event of an emergency. Minimizing this number is a critical factor for ensuring safety and reducing the social and political risks associated with nuclear energy production.
- **Degree of Operational Responsibility in Points:** This criterion measures the level of responsibility and operational safety associated with the plant's design and management. A higher degree of operational responsibility typically indicates a more secure and well-managed facility, contributing to both safety and regulatory compliance.

The evaluation of these three potential locations will provide a comprehensive understanding of each site’s strengths and weaknesses based on the established

criteria. The ultimate goal is to identify the most suitable location for the new nuclear power station, taking into account the balance between investment, operational efficiency, safety, and social impact.

Primary collected data is available in the table:

	criteria					
Alternatives	Number of workers	Power [MW]	Investment [mld. Kč]	Operating costs [mil. Kč]	Number of evacuated residents	Degree of operational reliability [points]
1	6500	4000	90	400	5500	9
2	4800	2400	50	300	4000	7
3	7500	4800	100	500	6000	8

1. Assign weights to the criteria

The expert panel will be assembled by four experts. The fuller triangle method will be used for evaluating the criteria.

Each expert receives a list of all pairs of criteria and marks the more important of them (one point) or marks both as equally important (half a point each).

The number of given points is written in the next table. To check the correct number of points we will calculate the checksum in every row. Having six criteria, expert must give 15 points (5 + 4 + 3 + 2 + 1):

	criteria						
Expert	Number of workers	Power [MW]	Investment [mld. Kč]	Operating costs [mil. Kč]	Number of evacuated residents	Degree of operational reliability [points]	checksum
E1	2	2	3	2	4	2	15
E2	3,5	2	4,5	3,5	1	0,5	15
E3	1	3	1	2	3	5	15
E4	3,5	2,5	3	1	1	4	15
Bj	2,5	2,375	2,875	2,125	2,25	2,875	

The importance coefficient B_j is calculated as the sum in the column divided by the number of experts.

2. Score the alternatives

To evaluate the alternatives we will use the base method.

In the first table we will identify the type of criteria and count the base as the average value of every criterion:

	criteria						
Alternatives	Number of workers	Power [MW]	Investment [mld. Kč]	Operating costs [mil. Kč]	Number of evacuated residents	Degree of operational reliability [points]	checksum
1	6500	4000	90	400	5500	9	15
2	4800	2400	50	300	4000	7	15
3	7500	4800	100	500	6000	8	15
Bj	2,5	2,375	2,875	2,125	2,25	2,875	15
type	-	+	-	-	-	+	
base	6266,67	3733,333	80	400	5166,67	8	

3. Rank the alternatives

The last step is the recounting of all values based on the type of criterion:

	Criteria							
Alternatives	Number of workers	Power [MW]	Investment [mld. Kč]	Operating costs [mil. Kč]	Number of evacuated residents	Degree of operational reliability [points]	Si	Order
1	2,41	2,54	2,56	2,125	2,11	3,23	14,98	2
2	3,26	1,53	4,6	2,83	2,91	2,52	17,65	1
3	2,09	3,05	2,3	1,7	1,94	2,88	13,95	3

Now we can count the usefulness of alternatives S_i and order alternatives. The highest value is the best.

4. Justification for decision

Do not forgot justify your decision. Explain why the selected solution was chosen, highlighting the most significant factors that led to this conclusion.

Consider providing a brief analysis between the different criteria and how they influenced your final ranking.

🕮 9.1.3

Project: A combination of different methods

The methods of multi-criteria analysis of variants have great potential in combination with other methods for the effective solution of very complex problems. Read the next article "Decision Support System for Evaluating Suitable Job Applicants", presenting a solution combining fuzzy-logic expert system with multi-criteria decisions analysis.

9.1.4

Final project

Define your own example and cover or solve it in the following space. The content and structure depend on tool and approach.

Consult with your classmates.

```
# part 1
```

```
# part 2
```

```
# part 3
```

```
# part 4
```

Bibliography

Chapter 10

10.1 Sources

🕮 10.1.1

Sources:

- Argoid. 5 Use Case Scenarios for Recommendation Systems and How They Help. [cit. 2023-07-11] Argoid, available on Internet https://www.argoid.ai/blog/5-use-case-scenarios-for-recommendation-systems-and-how-they-help.
- B. Thorat, P., Goudar, R. M. and Barve, S. (2015). Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. *International Journal of Computer Applications* [online]. 2015, **110**(4), 31-36 [cit. 2023-07-11]. ISSN 09758887.DOI:10.5120/19308-0760.
- Bede, B. (2013). *Mathematics of Fuzzy Sets and Fuzzy Logic. Studies in fuzziness and soft computing*, v. 295 (2013). ISBN: 978-3-642-35221-8.
- Farana, R. The usability of algorithms from graph theory in the field of multicriteria analysis. *In: 26th Annual International Conference Economic Competitiveness and Sustainability 2024 Proceedings*. Mendel University in Brno, 2024, pp. 60-64. ISBN 978-80-7509-990-7. https://doi.org/10.11118/978-80-7509-990-7-0060
- Farana, R. (2016). Use of graphs in tasks of multi-criteria analysis of variants. (in Czech) In: *XLI. Seminar ASR '2016 "Instruments and Control"*. Ostrava: VŠB-TU Ostrava, 22. 4. 2016, s. 47 – 53. ISBN 978-80-248-3910-3. http://akce.fs.vsb.cz/2016/asr2016/Sbornik-ASR2016.pdf.
- Frankeová, M., Farana, R., Formánek, I. & Walek, B. Fuzzy-Expert system for customer behavior prediction. *Advances in Intelligent Systems and Computing*. Volume 764, 2019, Pages 122-131. DOI: 10.1007/978-3-319-91189-2_13. 7th Computer Science On-line Conference, CSOC 2018; Zlin; Czech Republic; 25 April 2018 through 28 April 2018; Code 213719, ISSN: 21945357, ISBN: 978-331991188-5. Click here for full text.
- Fuzzy Logic - Inference System. https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_inference_system.htm.
- Fuzzy Logic Toolbox [cit. 2023-07-11]. Available on Internet: https://www.mathworks.com/help/fuzzy/index.html.
- GeeksforGeeks. (2023, January 24). Fuzzy logic: Introduction. GeeksforGeeks, available on Internet: https://www.geeksforgeeks.org/fuzzy-logic-introduction/.
- Hrnjica, B., Mušić, D. and Softic, S. (2020). Model-Based Recommender Systems. In: Al-Turjman, Fadi, ed. *Trends in Cloud-based IoT* [online]. Cham: Springer International Publishing, 2020, 2020-06-02, pp. 125-146 [cit. 2023-07-11]. EAI/Springer Innovations in Communication and Computing. ISBN 978-3-030-40036-1. DOI:10.1007/978-3-030-40037-8_8.
- https://github.com/MadScientist29/Movie-Recommendation-System-
- https://github.com/rcz7795/Movie-Recommendation-System/blob/main/Movie_Recommendation_System.ipynb

- https://medium.com/@saibhargavkarnati/movie-recommendation-system-using-machine-learning-8f6393d71c83
- https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/
- https://www.freecodecamp.org/news/how-to-build-a-movie-recommendation-system-based-on-collaborative-filtering/
- https://www.freecodecamp.org/news/how-to-build-a-movie-recommendation-system-based-on-collaborative-filtering/
- https://www.kaggle.com/code/aditihinge527/movierecommendationsystem-mlproject
- https://www.kaggle.com/code/chahinebenali/content-based-and-tensorflow-recommender-system
- https://www.kaggle.com/code/rounakbanik/movie-recommender-systems
- https://www.kaggle.com/code/stpeteishii/movie-recommendation-by-surprise
- https://www.kaggle.com/code/yunasheng/movie-magic-data-driven-recommendations
- https://www.kaggle.com/datasets/parasharmanas/movie-recommendation-system
- Introduction to recommender systems [cit. 2023-07-11]. Available on Internet: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.
- Ishizaka, A. and Nemery, P. (2013). *Multi-criteria decision analysis: methods and software*. Chichester: John Wiley, 2013. ISBN 9781119974079.
- Iterators (2021, July 15). Collaborative filtering in Recommender Systems: Learn all you need to know. Iterators, available on Internet: https://www.iteratorshq.com/blog/collaborative-filtering-in-recommender-systems/.
- Jain, A. and Gupta, Ch. (2018). Fuzzy Logic in Recommender Systems. In: Castillo, O., Melin, P. and Kacprzyk, J. ed. *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications* [online]. Cham: Springer International Publishing, 2018, 2018-01-11, s. 255-273 [cit. 2023-07-11]. Studies in Computational Intelligence. ISBN 978-3-319-71007-5. DOI:10.1007/978-3-319-71008-2_20.
- Klimes, C. (2011). Model of adaptation under indeterminacy. *Kybernetika*, Volume 47 (2011), Number 3, Pages 356 – 369. Click here for full text.
- Klimes, C. and Bartos, J. (2015). IT/IS security management with uncertain information. *Kybernetika*, Volume 51 (2015), Number 3, Pages 408 – 419. Click here for full text.
- Klimeš, C., Krajčík, V. & Farana, R. Proposal of Complex Software Applications. *Software Engineering Trends and Techniques in Intelligent Systems, Advances in Intelligent Systems and Computing*, vol. 575, 2017, pp. 53 – 61. DOI: 10.1007/978-3-319-57141-6_6. Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 3. ISSN 2194-5357, ISBN 978-3-319-57140-9 (WOS 978-3-319-57141-6). Click here for full text.

- Leskovec, J., Rajaraman, A. and Ullman, J. D. *Mining of Massive Datasets* [online]. Cambridge: Cambridge University Press, 2014 [cit. 2023-07-11]. ISBN 9781107077232. DOI:10.1017/CBO9781139924801
- Mamdani, E., Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic control, *Int. J. of Man-Machine Studies*, Vol. 7, 1975, pp. 1–13.
- Marlin, B. M. (2004). Collaborative Filtering: A Machine Learning Perspective.
- Mbaabu, O. M. (2020, December 22). *An overview of Fuzzy Logic System*. Section. Section, available on Intenret: https://www.section.io/engineering-education/an-overview-of-fuzzy-logic-system/.
- Mining of Massive Datasets. Chapter 9 Recommendation systems, DOI: https://doi.org/10.1017/CBO9781139924801.
- *Multi-criteria analysis: a manual*. Department for Communities and Local Government: London, 2009. ISBN: 978-1-4098-1023-0.
- Novák, V. (1995). Linguistically Oriented Fuzzy Logic Control and Its Design, *Int. Journal of Approximate Reasoning,* vol. 12, 1995, pp. 263-277.
- Novák, V. (2010). Genuine Linguistic Fuzzy Logic Control: Powerful and Successful Control Method, *Computational Intelligence for Knowledge-Based Systems Design*, Hüllermeier, E. and Kruse, R. and Hoffmann, F. (eds.), Springer, Berlin, 2010, pp. 634 -644.
- Novák, V. and Perfilieva, I. (1999) Evaluating Linguistic Expressions and Functional Fuzzy Theories in Fuzzy Logic, *Computing with Words in Information/Intelligent Systems 1*, L. A. Zadeh a J. Kacpryk (eds.), Springer-Verlag, Heidelberg, 1999, pp. 383-406.
- Ocelíková, E. *Multikriteriálne rozhodovanie*. Second edition. Elfa s r.o., Košice 2002. 88 s. ISBN 80-89066-28-3.
- Recommendation systems: Principles, methods and evaluation. https://www.sciencedirect.com/science/article/pii/S1110866515000341.
- Recommender system [cit. 2023-07-11]. Available on Internet https://en.wikipedia.org/wiki/Recommender_system.
- Recommender Systems For Business - A Gentle Introduction [cit. 2023-07-11]. Available on Intetnet: https://www.width.ai/post/recommender-systems-recommendation-systems.
- Ricci, F., Rokach, L. and Shapira, B. (2010). Introduction to Recommender Systems Handbook. In: Ricci, F., Rokach, L., Shapira, B. and KANTOR, P. B. ed. *Recommender Systems Handbook* [online]. Boston, MA: Springer US, 2011, 2011-10-05, pp. 1-35 [cit. 2023-07-11]. ISBN 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3_1.
- Roy, A. (2020, July 31). Introduction to Recommender systems- 1: Content-based filtering and collaborative filtering [cit. 2023-07-11]. Medium, available on Internet: https://towardsdatascience.com/introduction-to-recommender-systems-1-971bd274f421.
- Sayantini. (2023, May 14). *What is fuzzy logic in AI and what are its applications?* Edureka, available on Intenret: https://www.edureka.co/blog/fuzzy-logic-ai/.
- Springer US. (n.d.). Recommender Systems Handbook. SpringerLink: https://link.springer.com/book/10.1007/978-1-0716-2197-4.

- Takagi, T., Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. on Systems, Man, and Cybern*, Vol. 15, 1985, pp. 116–132.
- Walek, B., Pektor, O. and Farana, R. (2021). Decision Support System for Evaluating Suitable Job Applicants. *Mathematics*. 2021; 9(15): 1773. ISSN 2227-7390. Available on Internet: https://doi.org/10.3390/math9151773.
- Walek, Bogdan and Fojtik, Vladimir. A hybrid recommender system for recommending relevant movies using an expert system. Online. *Expert Systems with Applications*. 2020, year 158. ISSN 09574174. Available from Internet: https://doi.org/10.1016/j.eswa.2020.113452. [cit. 2024-02-27].
- Wikimedia (2023, April 5). Collaborative filtering. Wikipedia, available on Internet: https://en.wikipedia.org/wiki/Collaborative_filtering.
- Zadeh, L. A. (1965) Fuzzy sets, Information & Control, vol. 8, 1965, pp. 338-353.
- Zadeh, L. A. and Kacprzyk, J. (1992). *Fuzzy Logic for the Management of Uncertainty*, J. Wiley & Sons, New York 1992.

🕮 10.1.2

Statement regarding the use of Artificial Intelligence in content creation

This content has been developed with the assistance of artificial intelligence tools, specifically ChatGPT, Gemini, and Notebook LM. These AI technologies were utilized to enhance the text by providing suggestions for rephrasing, improving clarity, and ensuring coherence throughout the material. The integration of these AI tools has enabled a more efficient content creation process while maintaining high standards of quality and accuracy.

The use of AI in this context adheres to all relevant guidelines and ethical considerations associated with the deployment of such technologies. We acknowledge the importance of transparency in the content creation process and aim to provide a clear understanding of how artificial intelligence has contributed to the final product.

PRISCILLA

priscilla.fitped.eu