UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

FACULTY OF CONTEMPORARY SCIENCES AND TECHNOLOGIES

TITLE:

"RECOMMENDATION SYSTEMS FOR COMPUTER SCIENCE PUBLICATIONS"

MENTOR:

PROF.DR.XHEMAL ZENUNI

CANDIDATE:

MUSTAFA FEJZA

Tetovo, 2021

Declaration

I certify that I am the original author of this work.

Abstrakti

Qëllimi i tezës është prezantimi i një sistemi rekomandues të bazuar në përmbajtje i cili do të rekomandojë botime të shkencave kompjuterike për përdoruesit bazuar në preferencat e tyre. Publikimet janë nxjerrë nga konferenca "Sistemet Neurale të Përpunimit të Informacionit" (SNPI) që është një nga konferencat më të rëndësishme të mësimit të makinerive në të gjithë botën.

Sistemi do t'i ndihmojë përdoruesit të gjejnë përmbajtje më të shpejtë, më të lehtë dhe më efikase me qëllim të tejkalimit të mbingarkesës së informacionit.

Ekzistojnë lloje të ndryshme të sistemeve të rekomanduesve, por tre të parat që përdoren më shumë janë: filtrimi bashkëpunues,sistemi i bazuar në përmbajtje dhe sistemet hibride.Këto sisteme kanë modele të ndryshme që varen nga sasia dhe cilësia e të dhënave që ato përpunojnë.

Në këtë tezë kemi përdorur qasjen e bazuar në përmbajtje. Sistemi i rekomanduesve të bazuar në përmbajtje është një përmirësim i sistemit bashkëpunues të filtrimit. Ky model nuk kërkon vlerësimin e përdoruesit për artikuj të ndryshëm, përkundrazi rekomandimi krijohet bazuar në ngjashmërinë midis artikujve.

Për shkak të rritjes së shumë shërbimeve të internetit të ofruara nga kompani si Amazon, Facebook, Microsoft dhe shumë të tjera, sistemet rekomanduese morën një rritje dhe po shfaqen çdo ditë në jetën tonë.

Nën këto sisteme vendosen algoritme të ndryshme me qëllim sugjerimin e përmbajtjes së duhur dhe artikullit përkatës për përdoruesit siç janë filmat për të shikuar, ose blerja e produkteve etj. Në disa industri këto sisteme janë shumë kritike sepse ato prodhojnë sasi të mëdha të ardhurash për kompaninë. Kompanitë si Netflix kanë organizuar sfida me çmime të mëdha me qëllimin për të zbuluar mënyra të ndryshme për të përmirësuar algoritmin e tyre (ÇmimNetflix).

Abstract

The purpose of the thesis is to present a content-based recommender system that will recommend computer science publications to users based on their preferences. The publications are extracted from the conference “Neural Information Processing Systems” (NIPS)(Sejnowski, 2015)which is one of the most important machines learning conferences worldwide.

The system will help users find content faster, easier, and more efficiently to overcome the information overload.

There are different types of recommender systems but the top three that are most used are collaborative filtering, content-based and hybrid systems. These systems have different models that are depended on the quantity and quality of the data they are processing.

In this thesis, we used the content-based approach. The content-based recommender system is an improvement of the collaborative filtering system. This model doesn’t require the user’s evaluation for different items, instead, the recommendation is created based on the similarity between the items.

Because of the rise of many web services offered by companies like Amazon, Facebook, Microsoft, and many others, recommender systems received a boost and are appearing daily in our lives.

Underneath these systems lay different algorithms to suggest the right content and the relevant item to users such as movies to watch, or buying products, etc. In some industries, these systems are very critical because they generate large amounts of income for the company. Companies like Netflix have organized challenges with huge prizes to discover different ways to improve their algorithm (Netflix Prize).

Апстракт

Целта на тезата е да се претстави систем за препораки засновани на содржини кој ќе им препорача накорисниците компјутерски науки публикации врз основа на нивните преференции. Публикациите се извлечени од конференцијата „Системи за обработка на нервните информации“ (СОНИ) која е една од најважните конференции за машинско учење низ целиот свет.

Системот ќе им помогне на корисниците да најдат содржина побрзо, полесно и поефикасно со цел да го надминат преоптоварувањето на информациите.

Постојат различни типови на препорачани системи, но првите три што најмногу се користат се: колаборативно филтрирање, заснована на содржина и хибридни системи. Овие системи имаат различни модели кои зависат од количината и квалитетот на податоците што ги обработуваат.

Во оваа теза го искористивме пристапот заснован на содржината. Системот за препораки заснована на конкурентност е подобрување на системот за соработка на филтрирање. Овој модел не бара проценка на корисникот за различни артикли, наместо тоа, препораката се креира врз основа на сличноста помеѓу артиклите.

Поради порастот на многу веб-услуги понудени од компании како Амазон, Фејсбук, Мајкрософт и многу други, препорачаните системи добија поттик и се појавуваат секој ден во нашите животи.

Под овие системи се поставени различни алгоритми со цел да се предложи вистинската содржина и релевантната ставка на корисниците, како што се филмови за гледање, или купување производи и сл. Во некои индустрии овие системи се многу критични затоа што создаваат големи количини на приход за компанијата. Компании како Netflix имаат организирано предизвици со огромни награди со цел да откријат различни начини за подобрување на нивниот алгоритам (Награда Netflix).

Acknowledgments

First of all, I would like to thank my thesis advisor Prof.Dr. XhemalZenuni for his support and desire to give me the opportunity to work on this project. He was with me from the beginning, providing excellent support in many ways and showing keen interest in my project.

Special thanks to members of the council: Prof.Dr. Visar Shehu and Prof.Dr. BujarRaufi, who gave me the chance and confidence to complete this thesis.

I would like to express my gratitude to professors, staff members, and fellow students from South East European University for their support during my Master’s program.

In the end, I thank my family for supporting me during this long road of studies attending both bachelor’s and master’s programs at South East European University.

Table of Contents

Table of Figures

1 Chapter 1 - Introduction

During our lifetime what we buy or consume is influenced by some type of recommendation; whether that's a friend, family member, colleague, some reviews, or feedback on the website. When we buy a product on Amazon, the company always recommends us similar products to that product we bought. Even Facebook based on our friend list recommends to us people that we may like to add as a new friend.

With the expansion of the Web and the growth of e-commerce, a pressing emerged for providing recommendations. Users had difficulties finding the appropriate items due to the large variety of items that were offered by the websites. During this exponential growth, a range variety of choices were offered to the users, which lead to the diminishing of the client's prosperity and the expansion of their poor choices.

Recommender Systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user (Burke, 2002).

These suggestions are identified with various decision-making processes, for example, the sort of music to tune in, what sort of news to peruse, or what things to purchase.

These RS are focused on specific types of items (news, movies), and based on their design and model they generate useful and effective suggestions for those specific items.

Along these lines, we understand that providing decent suggestions to clients is very beneficial and crucial for the organization.

1.1 Background

Nowadays the term “Artificial Intelligence” (AI) it’s used very commonly. AI is a large field that enables the creation of different applications for specific purposes. Many industries use AI but as the top five, I will list: healthcare, education, marketing, retail, and e-commerce. We find AI in our laptops, smartphones and very soon it will be in our homes as well. Big companies are testing the so-called “smart home” where most of the devices will be connected to the Internet.

As we can see AI has and exponential growth and day by day it surrounds us. With the growth of AI, arise many questions from people that are confronted with it such as:

- What does “Artificial Intelligence” mean?
- How will it affect our daily lives?
- What are the advantages and disadvantages of AI?

Through using AI people and companies get benefits by reducing cost, optimizing their daily tasks. When users use AI devices, they enable the system to learn from their behavior and propose suggestions to make the tasks easier and more efficient.

In the era of the Internet where a large number of items are available online, the user can't inspect or compare the items with the hope of finding cheaper products or better quality.

Many companies invest a huge amount of resources into Recommender System to solve this issue and the challenges that arise from the expansion of Information Overload.

We can say that the main purpose of the Recommender System is to connect users with information that they need, which is one way will help the user to find valuable content. This strategy benefits both the user and the company and it is a win-win.

The main reason for our content-based recommender system is to test how efficient is the content-based model in finding relevant publications or research papers based on the input of the user.

1.2 Problem Statement

There is an explosion of knowledge which has led to exponential growth in the number of publications yearly. Users experience difficulties in finding favorite research papers from a large volume of research papers available on the Internet. With the increasing volume of information available on the internet, it is even more difficult for users to find the exact information of interest.

By trying to build a recommender system you will face several problems. During our development process we addressed the following:

- What kind of publication features can be used for the recommender system?
- How do we create a similarity between publications based on their attributes?
- How to calculate the similarity between two publications?
- How do we choose the content or the features of the publication?

1.3 Motivation

During my studies in Computer Science and Business Informatics, I got familiar with different topics in areas such as Artificial Intelligence and E-Commerce. I saw how big tech companies such as Netflix, Amazon, YouTube, and Google increase their revenue with the help of Recommender Systems. The reason why these companies care about Recommender System is money. With the help of such systems, they can deliver to users actual value, provide personalized content, and recommender many items. My motive emerged from these topics for the purpose of researching how these systems work and how they are created.

1.4 Research Objective

The main objective of this research is to present a content-based recommender system that will be able to analyze the content of the publication by analyzing its attributes and finding predictive methods for calculating and comparing the content of the two or more publications and find their similarities.

We will focus on what kind of features are useful for creating a model, which attributes should we drop, how to remove words from the content of the paper that are not useful for the model, and how to use content-based techniques to calculate the similarity between the publications.

2 Chapter 2 - Overview of Recommender Systems

In this section, we will give a brief introduction to recommender systems, the reason behind why they were created, what is their main purpose, what types of recommender systems exist, and how tech giants are solving their problems with the help of recommender systems.

2.1 Definition

What is a Recommender System? "A Computer program that recommends some sort of resource based on algorithms that rely on some sort of user model, some sort of content model, and some means of matching the two"(Dron, 2009).

Recommender System emerged in the mid of 1990s as a specific field that derived from research areas such as information retrieval, cognitive science, forecasting, and consumer modeling.

The reason behind their creation is to help clients in finding their way through huge databases and different catalogs, by suggesting relevant items to users while taking into account their preferences (tastes).

2.2 Types of Recommender systems

Due to different requirements, there are different types of recommender systems that are specific in their techniques and model when recommending items to users. The demand for recommender systems is growing due to the huge amount of data available. The most known types of recommender systems are content-based recommender systems, collaborative recommender systems, and hybrid recommender systems.

2.2.1 Content-Based Recommender Systems

The filtering methods are based on the attributes of the content to create an individualized recommendation. The content is displayed through features and can link these to the ratings of the users. By doing this, the systems learn from the user profile and display appropriate recommendations(Burke, 2002). This system has an algorithm that is able to build the user profile based on the features of the item the users rated. It combines the rating and behavior of users with the content of the item. It has an advantage over other systems because it is able to make recommendations if sufficient data is not available for that item. A content-based recommender system is used by Netflix. It helps in recommending movies to their clients. For example: if a client has watched a Fiction movie and gave a high rating, then he will get suggestions from the same genre.

2.2.2 Collaborative Recommender Systems

These systems are one of the most commonly used. These kinds of systems create the user profile based on the rating of the item by the user and then comparing that profile to other group user profiles. It recognizes the similarities between the profiles and based on that similarity recommends the item to the user. It filters items by using the opinion of other users. This type of system has been around near a decade, but its roots are from soothing that humans have been doing long ago “sharing opinions with others”.

For decades humans have been sharing an opinion on tastes discussing many things such as: what kind of food they have tasted, which movie they liked, etc.

With the help of the Internet, we are now able to go beyond word-of-mouth. Instead of talking to hundreds of individuals now, we can connect with thousands of users and develop a personalized view about a particular item based on the rating that all other users and by reading their shared opinion.

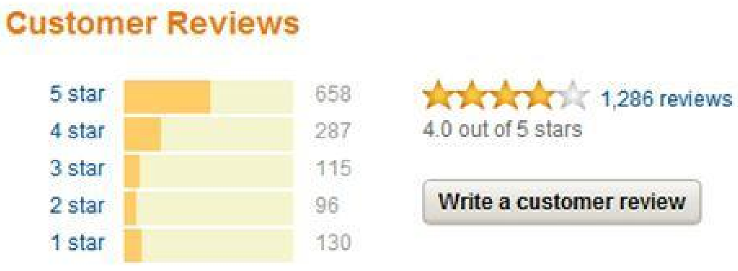


Figure 2.1- Types of ratings that collaborative systems use to recommend movies.

(Puleston, 2012)

2.2.3 Hybrid Recommender Systems

These systems are created by the combination of two or more recommendation strategies and their models to benefit from their complementary advantages. Researches have shown that by combining different models we can achieve better results. In such a hybrid approach the crucial part is the combination of the information given by the user and the content of the item. Six main techniques can be mixed for the creation of the hybrid system.

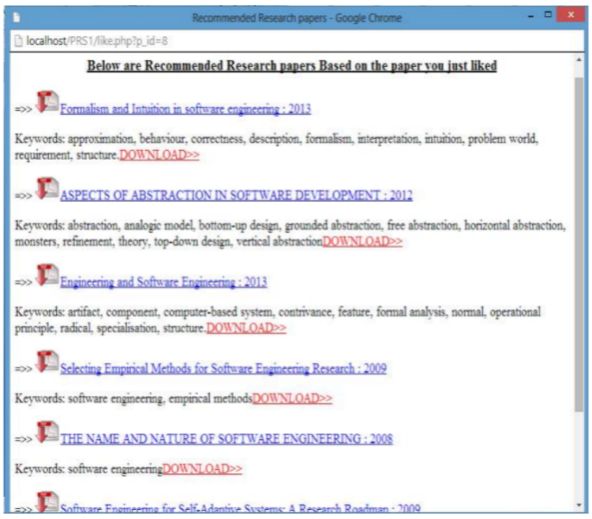
- Weighting: This technique uses the final score of the recommender systems and combines it to create a single recommendation.
- Switching: the system possesses many techniques and it can switch between them based on the current situation.
- Mixed: with the mixed model, different recommendations are used at the same time.
- Combined feature: in this technique, we select different features from several data sources and combine them into a single algorithm.
- Cascade: the recommender system refines the output that is given by other recommender systems. The aim is to optimize the results given by the recommenders.
- Feature augmentation: the output of the recommender system is used as an input for another recommender system.
- Meta-level: the purpose is to understand the meta structure of the problem and using the models that are learned by one recommender system as an input to another recommender system.

2.3 Usage of Recommender Systems

2.3.1 Paper Recommender System

Recommender Systems are becoming more and more useful in recent years. These systems are applied in different areas and applications. The most common areas are movies, news, music, research papers, and others.

Such a system was created by Simon Philip, P.B. Shola, and E.P. Musa were for the creation of the recommender system they used the content-based approach. This method was used for the design and implementation of the research paper recommendation systems that are based on the past ratings of an active user(Philip, (PHD), & E.P, A Paper Recommender System Based on the Past Ratings of a User., 2014).


Recommended Research papers - Google Chrome
localhost/PRS1/like.php?p_id=8

Below are Recommended Research papers Based on the paper you just liked

=>> Formalism and Intuition in software engineering : 2013

Keywords: approximation, behaviour, correctness, description, formalism, interpretation, intuition, problem world, requirement, structure.DOWNLOAD>>

=>> ASPECTS OF ABSTRACTION IN SOFTWARE DEVELOPMENT : 2012

Keywords: abstraction, analogic model, bottom-up design, grounded abstraction, free abstraction, horizontal abstraction, monsters, refinement, theory, top-down design, vertical abstractionDOWNLOAD>>

=>> Engineering and Software Engineering : 2013

Keywords: artifact, component, computer-based system, contrivance, feature, formal analysis, normal, operational principle, radical, specialisation, structure.DOWNLOAD>>

=>> Selecting Empirical Methods for Software Engineering Research : 2009

Keywords: software engineering, empirical methodsDOWNLOAD>>

=>> THE NAME AND NATURE OF SOFTWARE ENGINEERING : 2008

Keywords: software engineeringDOWNLOAD>>

=>> Software Engineering for Self-Adaptive Systems: A Research Roadmap : 2009

Figure 2.2- Paper Recommender System.

(Philip, (PHD), & E.P, A Paper Recommender System Based on the Past Ratings of a User., 2014)

2.3.2 Music Recommendation System

Here is another example where the recommender system was used for music recommendations. Why such systems are needed? Because music is emerging nowadays and users don’t have the time to search through many collations to find new items. These music recommendation systems are mostly created in two techniques: collaborative filtering or content-based. But, the system “Hybrid Content-Based Collaborative-Filtering Music Recommender”(Castillo, November 6, 2007)is created based on the hybrid method where both collaborative and content-based techniques are used.

2.3.3 Movie Recommender System

In 1997, Group Lens created the MovieLens recommender system (GroupLens, 1997)which was a non-commercial system that recommended films to users based on their preferences. The system was created using collaborative filtering techniques based on movie ratings and reviews. This system is based on the inputs provided by users. The system uses a variety of algorithms such as item to item(Sarwar, 2001)and user to user. For the cold start problem, they used surveys to ask users how much they enjoy watching a various groups of movies (horror, action, adventure). After the survey, the preferences are recorded and used by the system to make initial recommendations to its users without them rating movies.

2.3.4 News Recommender System

Bangla News Recommender System (Nandi, et al., 2018)was created using doc2vec. This unsupervised algorithm can generate vectors for documents or sentences. The algorithm is like an adaptation of the word2vec which generates vectors for words. The vectors that are generated from doc2vec can be used for finding similarities between documents.

3 Chapter 3 - Collaborative Filtering

These systems work by calculating the feedbacks that items receive from users in the form of ratings and exploiting similarities in rating behavior with other several users to determine how to recommend items. This method has two approaches: neighborhood-based and model-based.

3.1 Neighborhood-Based

Neighborhood-based collaborative filtering is one of the first algorithms that were developed for collaborative systems. The base of these algorithms is on the fact that there are users that have similar tastes, similar patterns, and similar behavior. Example:

1. Assume that we have a user marked as A and we weigh his rating.
2. We compare the weight of user A with other users.
3. Select users that have the same weight or are close to that weight.
4. We combine these weights and based on that we recommend an item to user A.

As we can see this algorithm uses weighting methods to measure the similarity between two users. The most common measure for the similarity between two users is the Pearson correlation coefficient.

$$W_{a,u} = \frac{\sum_{i\in I}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i\in I}(r_{a,i} - \bar{r}_a)^2 \sum_{i\in I}(r_{u,i} - \bar{r}_u)^2}}$$

Figure3.1 -Pearson Correlation Coefficient for weighting.

(I) = the items rated by both users. (ru,i) = the rating of the item (i) by the user (u), and (ru) describes the mean rating given by user (u).

- *Step 2:* after the weighting is computed we processed the second formula for predicting the item to the user:

$$\rho_{a,i} = \bar{r}_a + \frac{\sum_{u\in K}\left(r_{u,i} - \bar{r}_u\right) x\ W_{a,u}}{\sum_{u\in K} W_{a,u}}$$

Figure 3.2- Pearson Correlation Coefficient for prediction.

(pa,i) = the prediction for the active user (a) for item (i), (wa,u) = the similarity between users (a) and (u), and (K) describes the set of most similar users

- *Step 3:* we measure to the extent until there is a linear dependence between two variables.

$$W_{a,u} = \cos cos\,(\vec{r}_a, \vec{r}_u) = \frac{\vec{r}_a \cdot \vec{r}_u}{||\vec{r}_a||_2\, x\, ||\vec{r}_u||_2} = \frac{\sum_{i=1}^{m} r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}^2}\sqrt{\sum_{i=1}^{m} r_{u,i}^2}}$$

Figure 3.3 -Pearson Correlation Coefficient with cosine similarity.

It is important when computing cosine similarity, negative ratings don't exist and those items that don't have ratings are treated as zero.

Some of the extensions that were created to improve the performances of the neighborhood-based technique are:

3.1.1 Item-Based

When neighborhood-based collaborative filtering is applied to millions of users and items, the algorithms do not scale well, due to the complexity of the search for similar items. To improve this state, the item-based technique was invented where the method proposed the "item-to-item" technique. Instead of matching similar users, now they match the items that the user has rated to similar items. The advantages of this technique are a faster system and better results when recommending items.

$$W_{a,u} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

Figure 3.4 - Pearson Correlation Coefficient for comparing similarities between items.

U = all users who have rated item (i) and (j), (ru,i) = describes the rating of user (u) on item (i), r- = average rating of the (i) item across users

Now we use the weighting average technique to predict the rating for the item (i) to the user (a):

$$P_{a,i} = \frac{\sum_{j \in K} r_{a,j} W_{i,j}}{\sum_{j \in K} |W_{i,j}|}$$

Figure 3.5 - Prediction formula.

With (K) we describe the neighborhood items of (k) rated by (a) that are similar to (i).

During the implementation of the item-to-item method, instead of using Pearson correlation for similarities, we can use alternatives such as “adjusted cosine similarity”.

3.1.1.1 **Significance Weighting**

This method is created to multiply the similarity weight by a Significance Weighting factor that can devalue the correlations based on few co-rated items.

The reason this method was created is to avoid bad predictions. It was very common when a user was correlated with neighbor’s that were based on very few co-rated items.

3.1.1.2 **Default Voting**

Another method that helps in dealing with correlations based on few co-rated items is the default voting method. This method pleases a default value for the rating, for those items that have not received a rating. Now we can compute the correlation using the set of items rated by users being matched.

3.1.1.3 **Inverse User Frequency**

The items that received ratings from all users are not as useful as the less common items when measuring the similarity between users. For this purpose, Breese introduced the notion of inverse user frequency, which is computed as fi = log n/ni, where ni is the number of users who have rated item i out of the total number of n users. To apply inverse user frequency while using similarity-based CF we transform the original rating for i by multiplying it by the factor fi. The underlying assumption of this approach is that items that are universally loved or hated are rated more frequently than others(Breese, 1998).

3.1.1.4 Case Amplification

Was created to favor users that have high similarity to the active user.

$$W'_{a,u} = W_{a,u} \cdot |W_{a,u}|^{p-1}$$

Figure 3.6 - Case amplification. p = amplification factor

3.2 Model-Based

The way this model differs from the item-based technique is that it uses offline techniques to process the data. During the runtime, only the learned model is required to make predictions. It means we use fewer data to make a prediction. The item-based model uses all the data and is more reliable in predicting but has scalability problems, that’s why this model is invented to use fewer data.

There are four known approaches for model-based collaborative filtering:

3.2.1 **Clustering**

In this method, we divide users into groups knows, and clusters. The idea behind this divide is the assumption that users in the same group also have the same interest. This method uses one of the three formulas for measuring the distance between two users: the *Minkowski distance*, *Manhattan distance,* and *Euclidian distance*. The greater the distance the more dissimilar they are.

$$\frac{distance_{Minkowski(u_1,u_2)} = \sqrt[q]{\sum_j (r_{1j} - r_{2,j})^q}}{distance_{Euclidian(u_1,u_2)} = \sqrt[q]{\sum_j (r_{1j} - r_{2,j})^2} \; distance_{Manhattan(u_1,u_2)} = \sum_j |r_{1j} - r_{2j}|}$$

Figure 3.7- Minkowski, Euclidian and Manhattan distance.

The lesser the distance between (u1 – u2) the more similar they are.

3.2.2 **Classification**

Users are represented as rating vectors. Suppose every rating ranges from 1 to 5, and the user gives a rating of 5 to a particular item that means she/he likes such items. And the opposite if it gives a rate of 1 then she/he doesn't like that item. After the rating is done, each of the available ratings from 1 to 5 is considered as a class set. This technique uses the Bayesian method.

3.2.3 **Latent Class Model**

Assume we have user X and item Y. X and Y are a pair of user/item and are considered as a co-occurrence. Now, a latent class variable C is associated with each of the co-occurrences that appear. A set of latent classes is formed (C1 to CK). The problem that appears here is which latent class is more suitable to be specified to a co-occurrence. The scientists Hofmann and Puzieha used EX that stands for the "expectation-maximization algorithm" to estimate the probability.

3.2.4 **Matrix Factorization**

This method uses different techniques to analyze the rating matrix. It has two goals. The first goal is to reduce the dimension of the rating matrix, and the second goal is to discover features of the rating matrix.

This method uses some models such as Latent Semantic Analysis, Latent Dirichlet Allocation, and Singular Value Decomposition.

In the Collaborative Filtering model, two main problems appear the data sparseness and the huge rating matrix that are causing low performance to the recommender system. Not all the users and items contribute to predicting missing values, that's why they become unnecessary. This method aims to get rid of such users and to keep the most important one.

The usage of Principle Component Analysis (PCA) helps in dimensionality reduction. It finds significant components that are known as patterns. The patterns are the users.

3.3 Summary

There are different approaches to building a recommender system, but the collaborative approach is the most research one, because of its early use from the 90s where the systems were based on user communities that rated items, and nowadays most of the successful online recommenders rely on this technique.

The first systems were based on the memory approach and correlation-based algorithms. Later, various models emerged, such as the model-based approach where various techniques from different fields were used.

The availability of different test databases in many domains for the collaborative filtering model favored further the development of this system. Most of these databases are for movies (Movie Lense) and books.

Nowadays the Collaborative filtering techniques are very advance and are used in practical applications. But this system cannot be applied in every domain. Example: the car industry where no buying history exists, and where the ratings should be very detailed and cover all the details.

4 Chapter 4 - Content – Based Filtering

Until now we described the techniques that the collaborative filtering model is using. To make a recommendation, the techniques relied on user ratings, not on the item description. The main advantage of content-based filtering is the usage of the item to make a recommendation to the user. For example: in the real world if a friend has watched Fast and Furious 1, then we will recommend to him Fast and Furious 2 and some other movie with the same genre (action).

This type of recommendation is not available only with collaborative filtering. If we use only that model the model will recommend only Fast and Furious 2, no other action movies. For the second part of the recommendation, a content-based model arises. It will recommend a movie based on the same genre. The usage of both these models including the information that is available: the item and the user profile (ratings) forms a good recommendation system.

The content-based model doesn’t rely on many users to make a recommendation. It’s available to make a recommendation even if a single user is available.

Descriptions of the features and the characteristics of the item such as the genre of a book or movie, list of directors, actors, are often available in an electronic form. Challenging in this part are the qualitative features, in domains of quality and taste. For example, the reason a person likes something doesn’t rely always on the characteristics of the item itself (The Music Industry).

The content (features, attributes) that is used by the content-based recommender system is extracted from the metadata that is associated with the item. But this content that is extracted commonly it’s short and not enough sufficient to define the user’s interest. The usage of textual features helps in this part. Example: a content-based recommender system that recommends articles by comparing the keywords of both articles, the present with the past article that the user has rated.

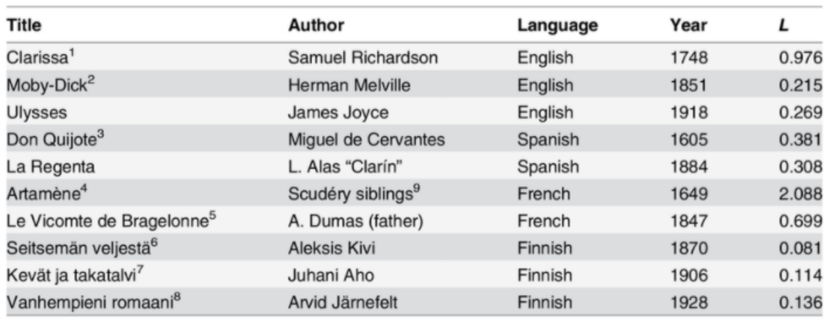
These kinds of techniques are known as knowledge-based approaches. Some authors say that the content-based approach is a subset of the knowledge-based model. But we will focus on the traditional classification where content-based are focused on exploiting the information from the item’s attributes, where eras knowledge-based recommenders use an additional utility function to produce recommendations.

Content-based filtering is more complicated than collaborative filtering, because of the challenges it faces when extracting knowledge from contents. To make a content-based recommender system we need:

- **Content Analyzer:** this technique creates a profile for each item. Here we train the model based on the content.
- **User Profiler:** creating a user profile. This profile can be just a simple list of the items that were consumed by the user.
- **Item Retriever:** in the last method we retrieve the items that were found when comparing the user profile with the item’s profile.

4.1 Content Representation

To describe items in a catalog the simplest way is to maintain a list of features for each of the items (known as attributes or characteristics). For example: to describe a book we can use the genre, author's name, publisher, year and store this information in a database. When the user will be described its preferences based on this set of features that the recommendation will be able to match the item's features with the user preferences.



Title	Author	Language	Year	*L*
Clarissa[1]	Samuel Richardson	English	1748	0.976
Moby-Dick[2]	Herman Melville	English	1851	0.215
Ulysses	James Joyce	English	1918	0.269
Don Quijote[3]	Miguel de Cervantes	Spanish	1605	0.381
La Regenta	L. Alas "Clarín"	Spanish	1884	0.308
Artamène[4]	Scudéry siblings[9]	French	1649	2.088
Le Vicomte de Bragelonne[5]	A. Dumas (father)	French	1847	0.699
Seitsemän veljestä[6]	Aleksis Kivi	Finnish	1870	0.081
Kevät ja takatalvi[7]	Juhani Aho	Finnish	1906	0.114
Vanhempieni romaani[8]	Arvid Järnefelt	Finnish	1928	0.136

Figure 4.1- Book characteristics.

(ResearchGate S. F., 2020).

How will this work? A user X will enter the author's name of the book or even the language that he prefers. Based on these inputs we will be able to check the database and recommend a book to the user or ask him for additional information such as the genre of the book. Then the system will derive a set of keywords from those types of books that the user X has liked in the past and recommend a book to the user. Also compares a not-yet-seen book with items that the user has liked in the past, to find similarities between them.

The similarity can be measured in different ways: genre, year, keywords. For example, a book that is not seen by the user can be compared with books that the user has liked in the past. If it's the same genre then the similarity is 1, if not then a 0.

Other options such as keywords are also useful. Measuring the books by their keyword's description. If a book is described by some set of keywords, then with the help of Dice Coefficient we measure the similarity between two books based on their keywords.

$$\frac{2\,x\,|\,keywords(b_i) \cap keywords(b_j)|}{|\,keywords(b_i)| + |\,keywords(b_j)|}$$

Figure 4.2- Dice Coefficient for measuring keywords.

4.2 Vector Space Model and TF-IDF

When speaking about the information of books, the author and publisher are not considered as the content of the book but as additional information. However, content-based systems have been developed for filtering news, e-mail messages. This means that the systems don't use the list of features such as author, publisher, but it uses the list of keywords that appear in a certain document.

There is a naïve approach where we can list all words that appear in all documents, and describe the document based on those words with 1 (if that word appears in the document) and 0 (the word doesn't appear in the document).

The problem with this approach is that every word in the document has the same importance, although a document that the word appears more often is better when characterizing a document. Also, long documents will be recommended more because a word will appear more often.

To solve the challenge, we use the TF-IDF encoding format. TF-IDF is a technique that comes from a field knows as "information retrieval" and has the meaning of "*Term Frequency – Inverse Document Frequency*".

Team Frequency is used to calculate how often a term appears in a particular document. We need to take into consideration the length of the document and prevent longer documents from getting higher relevance weight and normalize the length of the document.

$$TF_{(i,j)} = \frac{freq\,(i,j)}{maxOthers(i,j}$$

Figure 4.3 - Calculating the frequency and length.

The (i) describes the keyword, and (j) the document. (freq) the number of occurrences of the keyword (i) in the document (j). maxOthers stands for other words that appear in the document.

The second measure that is combined with team frequency is inverse document frequency. The aim is to reduce the weight of those words that appear too often in a document. The idea is to shift the weight of frequent words that are not helpful to give to those words that appear in only some documents.

$$IDF_{(i)} = \log log \frac{N}{n(i)}$$

Figure 4.4- Inverse document frequency.

(N) Describe all the documents in the recommender, (n) the number of those documents that keyword (i) appears.

4.3 TF-IDF versus Doc2Vec and BERT

4.3.1 TF-IDF

Is used as a statistical measure that can evaluate how relevant a word is to a document in a collection of documents. This is achieved by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents(Stecanella, 2019).

It is used mostly in text analysis and in Natural Language Processing for scoring words in machine learning algorithms. The purpose of inventing this method was the need for document search and information retrieval.

TF-IDF is calculated by counting how much a particular term appears in a document and the inverse document frequency of a word which measures the importance of that term in the set of documents. We take into account also that the longer the document the greater the possibility of a term appearing more times that’s why TF is calculated: the number of times a term appears in a document/ the total terms in that document. IDF calculates the total number of documents / the number of those documents that have that particular term.

4.3.2 Doc2Vec

This method was presented by Mikilov and Lee (Mikilov & Le, 2014). It is based on the word2vec (Mikolov, Chen, Corrado, & Dean, 2013) that is used for generating representation of vectors out of words.

The main purpose of doc2vec is to create a numeric representation of the document, regardless of its size or length. The difference between word2vec and doc2vec is that doc2vec has one more vector (Paragraph ID). With this method, we give documents unique keys and we don't train only the words vector, but also the document vector. For example, the word vector represents the concept of the word, and the document vector represents the concept of the document. The doc2vec generates a vector W for each word and a vector D for each document.

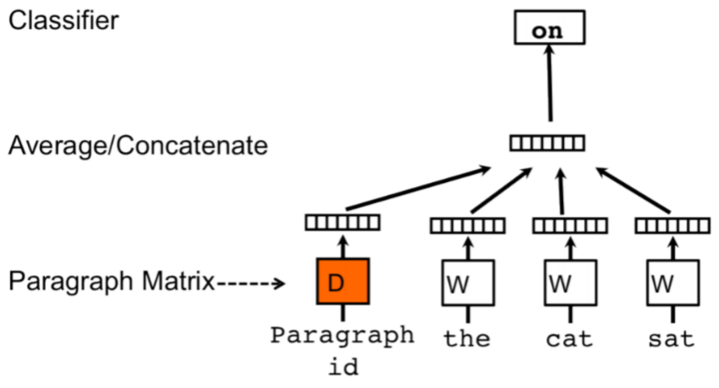


Figure 4.5 - Doc2Vec.

(Mohanty, 2020).

4.3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin, Chang, Lee, & Tautanova, 2018), it presented state-of-art to the Machine Learning community by showing great results in different NLP tasks, such as Questions Answering, Natural Language Inference, and others.

Bert can learn the contextual relations between words in a text. It has a Transformer that is separated into two mechanisms: an encoder that is used for reading the input, and the decoder that produces a prediction for the task(Horev, 2018).

The transformer encoder can read the entire sequence of words at once. It does not read like others from left to right or from right to left.

Both BERT and doc2vec are created by Google, to handle large data. BERT is a breakthrough in the Machine Learning field. With its approachable style and fast fine-tuning, it will lead to a variety of practical applications in the future. BERT code and model is now released by the research team and it is open source(Google, 2018).

4.4 Improving – Vector Space Model

Typically, such vectors as TF-IDF are very large and sparse. To make them more efficient and compact we need to remove irrelevant information.

4.4.1 Stop Words

The straightway to reduce the large vector is by removing the stop words. These stop words include words such as: “the”, “on”, “a”, etc. These words should be removed because they appear in every document.

Stemming is also a useful technique. It replaces a word with its root word. (went = > go). Stemming procedures are implemented as a combination of morphological analysis using Porter’s algorithm (Porter 1980) and with the help of the WordNet dictionaries. Using these techniques AI improved, but when using syntactic suffix stripping the risk still exists because the danger of matching a profile with a wrong document is high. For example, words such as universal and university are stemmed from the universe.

The usage of these techniques reduces the vector size and at the same time improves the matching process.

4.4.2 **Size Cutoffs**

Another method that is quite useful in reducing the size of the vector is by using the most informative words. A study was made in several domains about important words where the usage of only 50 keywords didn’t cover all features of the document. When using 300 keywords the word itself has limited importance and lowers the accuracy of the recommender system.

4.4.3 **Phrases**

The usage of “phrases as terms” does o further improvement to the recommender system. These are more descriptive for a text rather than a single word alone. Such phrases or composed words can be: “United Nations” and they can be encoded as dimensions in the vector space. The detection of these phrases can be done by applying statistical analysis techniques(Ricci, Rokach, Shapira, & Kantor, Recommender Systems Handbook, 2011).

4.5 Limitations

The approaches such as: extracting and weighting keywords from texts have an important limitation. It does not take into consideration the context of the keyword and in some cases, it can have problems catching the meaning of the description. For example, a restaurant that serves the only steak (steakhouse) may have in their menu description the following “this menu does not include foods for vegetarians”. In this case, the vector that will be generated will give a huge weight on the word vegetarians and will produce a match with the user that is searching for a vegetarian restaurant.

4.6 Similarity-Based Retrieval

Items in collaborative filtering recommender systems are described as “items that are similar users liked”. In content-based recommender systems, the description is “items that are similar to those items that the user liked in the past”(Patel, 2013).

As we can see the job of the recommender systems in both models is to recommend an item that the user has not seen, with the hope of them liking it.

4.6.1 Nearest Neighbors

How can we know that a certain document will be in the interest of the user? A simple method is by checking if he has liked similar documents in the past.

To implement this method, we need two pieces of information: a history of what the user has liked/disliked in the past, and a measure that captures the similarity between two documents. The first one can be provided either by the user interface or by monitoring the user's behavior. The second one is done with the help of cosine similarity for measuring the similarities between two documents.

For example, we have a pack of 5 documents that are similar. If the user has liked 4 of them, then the possibility of him liking the fifth one is very high.

The method is known as kNN (k-nearest-neighbor). Such methods were used starting from 2000 in news recommendations where the short-term interest of the user was very important(Yang, Adeniyi, & Wei, 2016). When a new story arrives, the system is able to check if the story is as similar as those the user has liked in the past, or if it's a follow-up story to a recent event then it will recommend it to the user. Also, when implementing such methods, the system will be able to prevent documents or news that the user has seen already.

This method was implemented as a part of a broader system for the user profile technique. The system induced both: short-term interest and long-term interest of the user(Billius & Michael, 2000). The short-term technique provides the user information's on topics of recent interest. For the long-term technique, the model collected information about the user over a long period of time (months, years) and was able to identify informative words in documents with the help of TF-IDF and recommend those documents or news that had the highest score in the collection.

The important thing in this method is to combine both the short-term and the long-term interest. A simple solution was proposed: first, we do a search on the short-term model: if a document (neighbor) was not found, then the systems continue to the long-term model.

The kNN method has many advantages: it is adaptive, is simple to implement, and is able to make recommendations based on a small number of ratings. But if we compare kNN with other sophisticated techniques we receive a lower score.

4.6.2 Relevance Feedback – Rocchio's Method

This method is also based on the vector-space model. During a search for an item by the user, despite entering the keywords they were able to give feedback (rate) of the items they received whether those items were relevant. Based on the feedback the system could improve the retrieval results in the upcoming searches. This system is known as the SMART system.

The system did not exploit additional information, but it allowed the user to interact with the document (rate it) and tell the system whether the result is good or bad.

Why this approach is needed? Without the feedback, the systems will only rely on the individual's capability and the quality of the keywords that they will input. The typical search query on the web consists of only two keywords on average.

After the document is rated by the user, the main idea is to put those documents into two groups: G+ (the documents that the user liked) and G- (the documents that the user disliked). Then, calculating the average vector of these two categories.

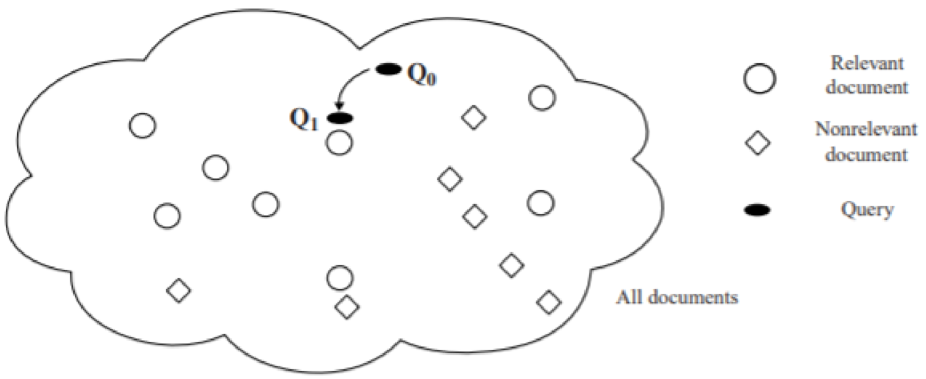


Figure 4.6 - Two groups and the search query.

(Manning, 2008)

After receiving a good feedback, the document is moved towards the group of relevant documents.

Nowadays the feedback method is used in various domains. It is has shown that despite its simplicity it can lead to improvement and better performance of the recommender system. But, to build a good model, the system will rely on a certain number of ratings and user interactions.

The gathering of ratings can be done in many forms. For example: if a user is recommended with 5 documents, a click on one of those documents can be calculated as a positive rating. The problem part is what to do in scenarios when the user has read the document and was disappointed.

Another technique is to rely on the so-called blind feedbacks. Example: assume the user got a recommendation of 10 documents. If none of those documents received a bad review then we consider them relevant items, without the user giving them positive feedback, but in the idea that they didn't receive either bad feedback.

The point is to learn the desired preferences of the users with their low interaction. Nowadays web users are very impatient and are not willing to give feedback on recommended items.

Content-based recommender systems that are based on the query approach are very similar to modern search engines (Google). Such tech giants like Google, Yahoo offer many other features (email, document management, and manipulation), and based on these features they can view the user’s behavior and learn from it, to improve their recommender system.

4.7 Probabilistic Methods

These methods are based on classes such as “like” and “dislike”. When formulating a recommender system as a classification problem then various techniques can be applied from the supervised machine learning techniques.

Such approaches are based on the naïve Bayes assumption of conditional independence and have been successfully deployed in content-based recommender systems.

$$P(B) = \frac{P(A)P(A)}{P(B)}$$

Figure 4.7- Naïve Bayes classifier.

(A|B) is the probability of A being true given that B is true, (B|A) is the probability of B being true if A is true, (A) probability of being true, (B) the probability of being true

Classifiers are also used in the collaborative-filtering recommender systems to determine the membership of the user into a cluster of users that has the same preferences, where eras in the content-based approach the classifiers are used to determine if a certain document is in the interest of the user.

When using the naïve Bayes classifier, we need to take into consideration that not every individual event is conditionally independent because there are co-occurrence terms such as New and York (New York) or Hong and Kong (Hong Kong).

Despite these limitations in co-occurrence words, the Bayes classifier has shown it is useful in text classification.

The Bayes classifier despite its good accuracy is also able to improve because the components of the classifier can be easily updated and can absorb data when they are available. However, as with all other techniques, a certain number of training data (ratings) is required to provide precise recommendations.

The cold-start problem is also present in such content-based approaches where some sort of feedback is needed. There are ways to deal with this, such as; asking the user to provide a list of words for each category.

4.8 Other Classifiers

When we view the content-based recommender system as a classification problem, then various other techniques appear and are available to be implemented. The purpose is to find a linear model to discriminate documents that are relevant and non-relevant.

There are sketches where the documents are characterized by just two dimensions. If only two dimensions exist, then the classifier can be represented with just one line.

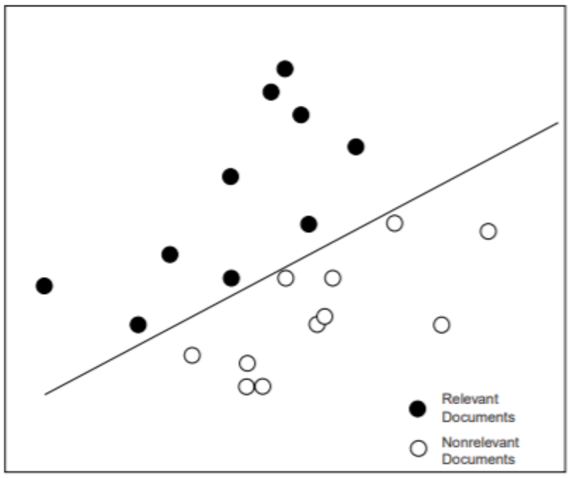


Figure 4.8 - A classifier with one line and two dimensions.

(ResearchGate, 2008)

Various challenges can appear when using such linear classifiers. For example noise documents. This challenge appears when there are noisy features that can mislead the classifier. Because of these features’ documents may not be near the cluster they belong to.

4.9 Explicit Decision Models

There are two other techniques: decision trees and ruled induction; which are used for building a content-based recommender system. These techniques are different from other techniques because of the generation of explicit decision models in the training phase.

Such techniques are also useful in other domains such as data mining. When they are used in the recommendation system, the features of the item are labeled with the inner nodes of the tree, and the nodes are used for the partition of the text examples, for example: for the existence or nonexistence of a keyword in a document.

The determination of whether a document is relevant can be done in a very efficient way with such classification trees, which can be constructed from the training data, without the need of using domain knowledge. Decision trees are well understood, can be applied in various domains, and are easy to be interpreted.

The main issue when applying such techniques (decision trees) lies in the usage of many features. For example: in content-based recommender systems we use TF-IDF for document representation. Decision trees are useful when there are small numbers of features, in which case we should be using meta-data features such as name, genre, and not use TF-IDF for document representation.

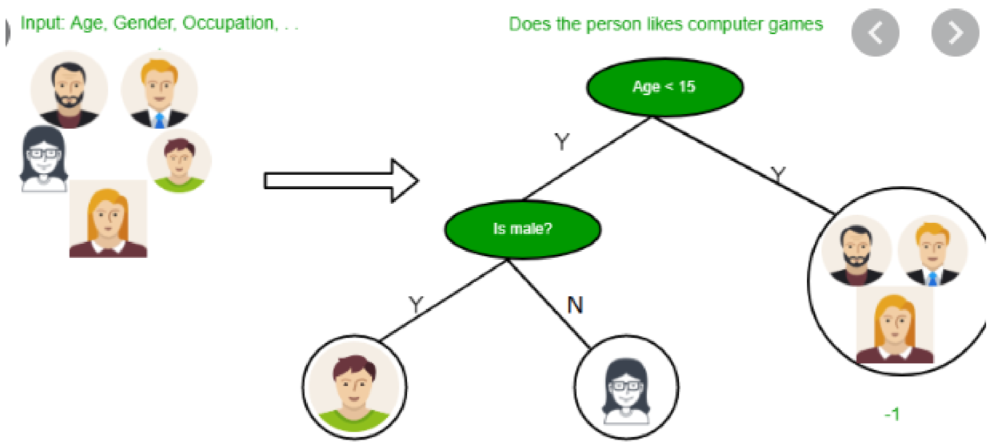

Input: Age, Gender, Occupation, . .
Does the person likes computer games
Age < 15
Y
Y
Is male?
Y
N
-1

Figure 4.9 - Decision Tree example.

(Geeksforgeeks, 2019)

The reason for this limitation lies in the splitting strategy of the decision tree. Due to large information, it can lead to small decision trees (Pazzani, Muramatsu, & Billsus, 1996).

For this reason, content-based recommender systems that are created based on the decision tree model are used in classical scenarios and are used in combination with other techniques and not as a core technique. Example: to determine with user model features are most relevant for providing better accuracy or to compress in-memory data structures.

Rule induction is also a method that is used for extracting decision rules from the training data. This method has been used for e-mail classification.

Both of these methods; decision tree and rule induction are very successful and have been used in subproblems such as e-mail classification, for advertisement personalization, or even in some cases where the item has just a few features. As we described previously these techniques have two main advantages: decision rules can serve for generating explanations for the recommendations and other domain knowledge can be incorporated into the models.

4.10 Feature Selection

These techniques that we described previously, rely on the vector representation and in the weights of TF-IDF. Document vectors if used in a very straightforward way tend to be very large with thousands of words appearing in the corpus, even with the removal of the so-called "stop-words" and the steaming technique.

When used in applications, such large vectors cause problems in many areas: slow performances, memory full, and the effect of overfitting. For example, we may have documents labeled as "hot". During the training phase, this noisy data will lead the classifier into describing these documents as very interesting for the user. Overfitting appears when only a small number of training documents are available.

To avoid such problems, it is desirable to use only a subset of all the terms that appear in the corpus of classification. This process is known as “feature selection”. The main idea is to choose just some features and not all of them.

The removal of irrelevant features can lead to better results and accuracy. Other techniques and strategies are the removals of those words that are too rare or that appear too often into a document.

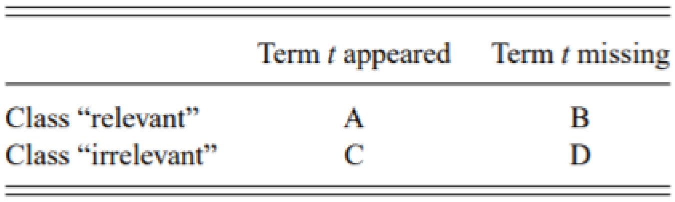
How do select features that will improve the accuracy of the recommender system?

The optimal feature can be found by training the classifier on every feature and evaluate its accuracy. After the implementation, a list should be formed of “good” keywords. The measurement of determining such keywords is done by the X2 test.

$$x^2 = \frac{(A+B+C+D)(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)}$$

Figure 4.10- X2 test.

The x2 test is a standard statistical method that is used to check if two events are independent. Based on this idea, in the context feature, we should analyze into the training data, whether classification outcomes are connected with specific term-occurrences. If a dependency for a term is identified, then we will include this term into the vector for classification.



	Term *t* appeared	Term *t* missing
Class “relevant”	A	B
Class “irrelevant”	C	D

Figure 4.11 - Contingency table of x2.

(Jannach, Zanker, Felfering, & Friedrich, 2011)

(t) Is the term. (A) Describes the number of documents that contain the term “t”. (B) Describes those documents that are relevant, but don’t have the term (t). (C) and (D) describe all those documents that are irrelevant.

The creation of contingency tables enables the system to check if a term is relevant or irrelevant to a document.

Fisher’s discriminant is also a technique that is used in information retrieval.

4.11 Limitations

The content-based recommender system has its limitations. This led to the creation of the hybrid-based recommender system that is created by using advantages of other recommender systems and their models.

Example of limitations:

- Recommending web pages to users. The problem that appears in this scenario is that analyzing a web page only by its content is not enough. We need to consider the usability, check hyperlinks, the quality of the page, and others.
- In other scenarios where keywords are used for the characterization of documents, the recommender system is limited to determine whether the document is a well-written article or a poor one.
- There are also hypertext documents that have multimedia elements (images, audio, and video). These contents are not considered and not taken into account by the content-based recommender system. The research in the extraction of these features is still in an early stage.
- These recommender types tend to propose more of the same items, such as: in a newspaper if an event has happened, the user may already read an article for that event, but the system will recommend other news for that same event that is created in a different context. To avoid this problem the system should increase the serendipity of the items, to include some random items that the user might be interested in.

- The cold-start problem is also present in the content-based approach. Before the collaborative approach that requires a lot of ratings, here we need at least a small portion of ratings.

Challenges exist. Most of the learning techniques require a certain amount of training data to achieve better recommendations. The border between content-based recommender systems and other system is not strictly defined. Pure content-based recommender systems are rarely found in the e-commerce industry. In many cases, they are used in combination with other systems.

5 Chapter 5 - Hybrid Recommender Systems

The previous two recommender systems that we described: collaborative and content-based follow different paradigms to make recommendations. They produce good results that are personalized based and perform successfully in various domains. For example collaborative model uses the item ratings and the community data to make recommendations, the content-based model relies on the features of the items and textual descriptions. Each of these models has its pros and cons, starting from the data scarcity, to the cold-start problem and others. However, none of these models is great enough to perform on its own with good results in the real-world.

To overcome such problems and limitations, and to be practical and useful in the real world, researchers created a new model that is known as the hybrid system. This system is created by combining the strengths of other models. Even the word “hybrid” has its origin in the Latin noun “hybrida” which stands for mixed origins.

As we mentioned previously, a hybrid recommender system is a combination of several algorithms.

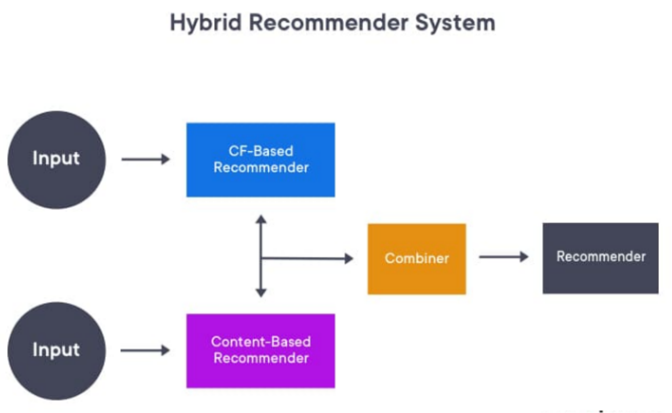


Figure 5.1 - Hybrid recommender system.

(Singla, 2019)

Most of the practical recommender systems that exist are hybrid-models, but in the theoretical approach, not a lot of work has been focused on how to combine different algorithms. The Netflix prize was such an example where to increase the accuracy of the system, they included different algorithms into the system.

When forming such hybrid systems, the task of the collaborative model is to determine similar items to the user by looking into their rating history, and the content-model will be recommended much of the same by looking into the features of the items that the user has liked.

5.1 Hybridization Design

Hybrid systems are very unique in their design. We will discuss three based designs: pipeline, monolithic, and parallelized hybrids.

5.1.1 **Monolithic**

To create such a design, we need to incorporate several recommendation strategies into one algorithm. Every strategy contributes in its way when analyzing the data. The input is analyzed by the hybrid system that is formed from different strategies and outputs the results (Figure 5.2).

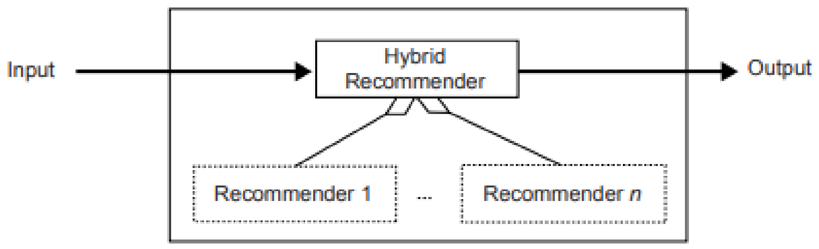


Figure 5.2- Monolithic design.

(Jannach, Zanker, Felfering, & Friedrich, 2011)

5.1.2 **Parallelized**

This design requires two separated recommender systems which are combined into a bigger system. Based on the input these recommenders receive they work independently of one another and produce separated results.

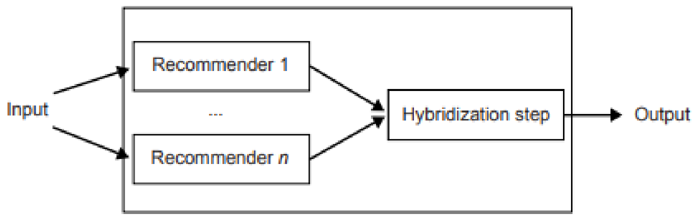


Figure 5.3- Parallelized design.

(Jannach, Zanker, Felfering, & Friedrich, 2011)

5.1.3 **Pipelined**

In the pipelined design when two recommender systems are joined together, the output of one recommender helps as an input for the other recommender.

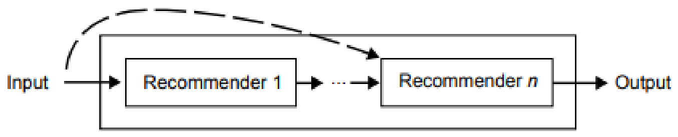


Figure 5.4- Pipelined design.

(Jannach, Zanker, Felfering, & Friedrich, 2011)

5.2 Hybridization Techniques

As we discussed above, there is three main design for building a hybrid system where two of those design is created by two or more recommenders and just one of them; the monolithic design is built by just one recommender. This design is created via feature combination or feature augmentation.

5.2.1 Feature Combination

Is a monolithic recommendation component that for input uses a diverse range of data. Example: combines collaborative features (likes and dislikes of users) with content features (features of items).

5.2.2 Feature Augmentation

Is also a monolithic design that does not simply combine and process several inputs, but can take it to the next level by applying complex transformation steps. The output of one recommender is used as an input for other recommenders. In this method, the input it receives from other recommendations treats it as a feature. It has similarities with the stacking technique that is used in a classification where the outputs of one classifier as used as features for the next one.

5.2.3 Weighting Average

The weighting average technique aims to find the weights that have results and are very accurate. These weights are from the outputs of content-based and collaborative filtering models. In some cases, the system weights both models equally. When both of the models recommender the same item, then that item is recommender to the user.

5.2.4 Switching

In this technique, the algorithm can switch between different recommender systems based on the current needs. For example: in the early phase it can use the content-based approach to avoid the cold-start problem, then in the late phases where ratings will be available it can switch to the collaborative model. Also, the system will pick the recommender system that will provide the most accurate items at a given point in time.

5.2.5 Cascade

This technique refines the recommendations given by another recommender system. The training process of one recommender system is biased by the output of other recommenders, and the overall results are combined into one output.

5.2.6 Meta-level

In this technique, the model used by one recommender system helps as an input for another recommender system. A typical example is the combination of a content-based and collaborative system. Here, the collaborative system is modified so it can be able to use the content of features for the determination of peer groups. Then with the rating matrix is able to recommend items and make predictions. Note, the collaborative system first uses the modified version of itself by analyzing the content features to determine the peer groups, and only, in the end, it uses the rating matrix for a recommendation. These systems are known as “collaboration via content”.

5.2.7 Mixed

This technique presents to the user recommendation from several models at the same time. This system doesn’t combine the scores from various recommenders as other techniques do.

These recommenders are considered neither monolithic nor ensemble-based, but they are classified as a category of their own. This model is mostly used in cases where the recommendation system is a composite entity where multiple items are recommender as a related set.

5.3 Summary

Hybrid recommender systems are used for two main reasons: to leverage the power of multiple data sources and to improve the performances of recommender systems. The motivation to create and evolve such a recommender system is the desire to use the strengths of content-based and collaborative filtering systems. Some of them were good in the cold-start problem, and some when more data was available.

6 Chapter 6 -Building the Recommender System

This section will give an overview of the system, how it was created, the system framework and requirements, and basic conditions.

6.1 System Requirements

During the creation of the system, we used the Anaconda Environment (64 –bit) with Python version 3.7. Anaconda is created by Continuum Analytics, and it is used a lot by Data Scientist because it is a Python distribution with lots of libraries available.

This environment is popular because it has a lot of tools that are used in data science and machine learning. It has its package manager known as “conda” where we install the libraries, but the “pip” manager is also available.

Recommended requirements:

- Processor: Intel ® Core ™ i3-3220 CPU @3.30Ghz
- RAM: 8.00 GB

- 64-bit Operating System
- Storage: 10GB +.
- Operating System: Windows 10.

6.2 Libraries

For the creation of the content-based recommender system we used different software libraries, such as:

6.2.1 **Pandas**

It is a library that is used for the manipulation and analysis of data. It is written for the Python programming language it is used like Excel, with tables knows as Data Frames, it has rows and columns known as Series, and many more functionalities that help in data processing and manipulation. This library relies a lot on NumPy. Pandas is already pre-installed in the Anaconda environment, but without this environment, we can install it in CMD with the following command: “pip install pandas”

6.2.2 **NumPy**

It is a powerful open-source library. It’s used a lot for array computing and metrics. It has a large collection of mathematical functions that can operate with the arrays. It’s one of the most used libraries in python along with Pandas.

6.2.3 **Re**

Re is known as a regular expression and it is used to identify patterns in a sequence of characters. Mainly is used for the manipulation of textual data, in the field of text mining, pattern identification, and input validation.

6.2.4 **Plotly**

It is an open-source library, interactive with the main purpose of creating visual data. Supports 40 unique chart types, starting from statistical, financial, scientific, and 3D cases. We can create beautiful web-based visualization that can be displayed in the Jupyter Notebook.

6.2.5 **Cufflinks**

It is a third-party library that works with Plotly. When we use cufflinks all the data frames in Pandas have a new method known as iplot.

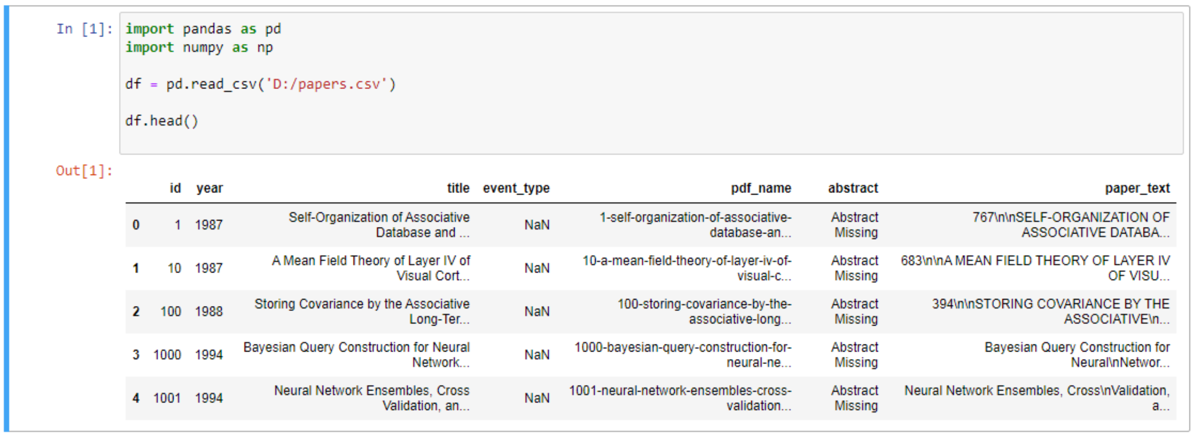
6.2.6 **NLTK**

Stands for Natural Language Toolkit, and it is created from different libraries of Natural Language Processing. It is used mainly for: converting text to lower case, word tokenization, removal of stop words, word frequency, and others.

6.3 Dataset

NIPS dataset is used as data for the system to generate. Neural Information Processing Systems (NIPS) is one of the top machine learning conferences in the world. It covers topics ranging from deep learning and computer vision to cognitive science and reinforcement learning.

This dataset includes the title, authors, abstracts, and extracted text for all NIPS papers to date (ranging from the first 1987 conference to the current 2016 conference). I've extracted the paper text from the raw PDF files and am releasing that both in CSV files and as an SQLite database(Hamner, 2017).



```
In [1]: import pandas as pd
        import numpy as np

        df = pd.read_csv('D:/papers.csv')

        df.head()
```

Out[1]:

	id	year	title	event_type	pdf_name	abstract	paper_text
0	1	1987	Self-Organization of Associative Database and ...	NaN	1-self-organization-of-associative-database-an...	Abstract Missing	767\n\nSELF-ORGANIZATION OF ASSOCIATIVE DATABA...
1	10	1987	A Mean Field Theory of Layer IV of Visual Cort...	NaN	10-a-mean-field-theory-of-layer-iv-of-visual-c...	Abstract Missing	683\n\nA MEAN FIELD THEORY OF LAYER IV OF VISU...
2	100	1988	Storing Covariance by the Associative Long-Ter...	NaN	100-storing-covariance-by-the-associative-long...	Abstract Missing	394\n\nSTORING COVARIANCE BY THE ASSOCIATIVE\n...
3	1000	1994	Bayesian Query Construction for Neural Network...	NaN	1000-bayesian-query-construction-for-neural-ne...	Abstract Missing	Bayesian Query Construction for Neural\nNetwor...
4	1001	1994	Neural Network Ensembles, Cross Validation, an...	NaN	1001-neural-network-ensembles-cross-validation...	Abstract Missing	Neural Network Ensembles, Cross\nValidation, a...

Figure 6.1 - The dataset.

As we can see in (Figure 28) the dataset has 7241 rows and 7 columns. For the recommendation system, we are going to use the id, title, and paper_text. Other columns such as event_type, year, pdf_name, abstract, will be dropped because event_type and abstract have no data, and year with pdf_name are not useful for us because we will not search based on the year or the pdf name.

6.4 Data Cleaning

Cleaning the data is very important for a Data Scientist. According to IBM Data Analytics (Analytics, n.d.) The data scientists spend 80% of their work cleaning the data. To build the Matrix we will use the paper_text column, that's why it's important to clean that column.

The "Stop Words" method will be used. Stop Words are those words that in the field of natural language have little meaning, such as: "an", "is", "the", and others.

Stop Words are removed before we create the model, because it will occur in abundance, and will provide very little information if used for classification.

There are different libraries for removing stop words such as NLTK, SpaCy, Gensim, and others. We used the NLTK library for our system.

NLTK is one of the oldest libraries, and it is used very commonly. This library works in a way that it divides the text into words, and if a word exists in the NLTK list, it will remove it.

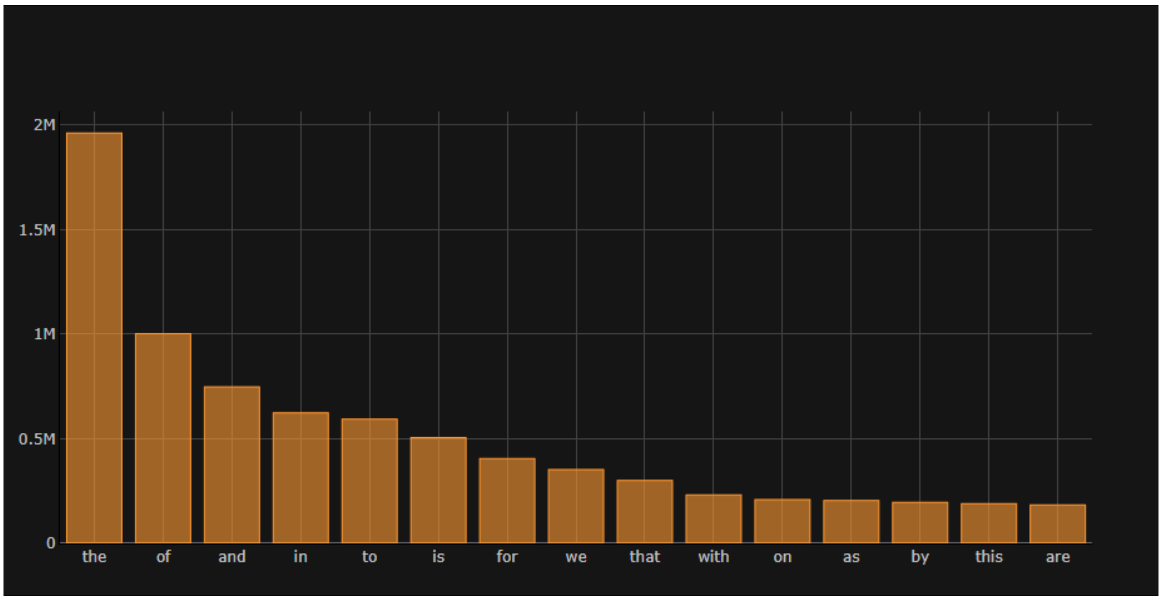


Figure 6.2 - Most used words in column paper_text before using stop words.

Figure 6.2 describes the top fifteen words used in NIPS Papers. We see that these stop words such as the, of, and, in, to, are used a lot. The word "the" is used 1.9 million times.

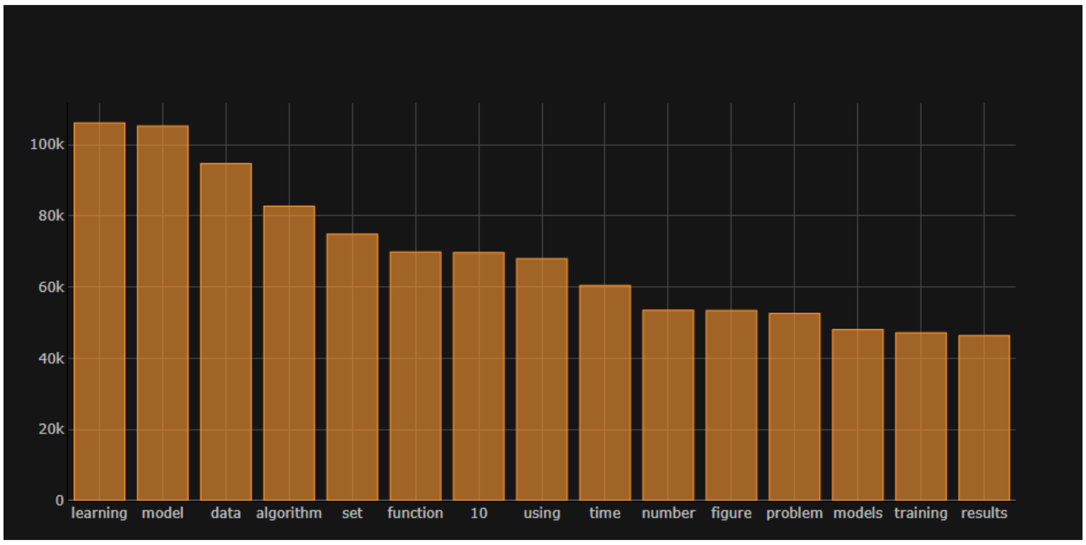
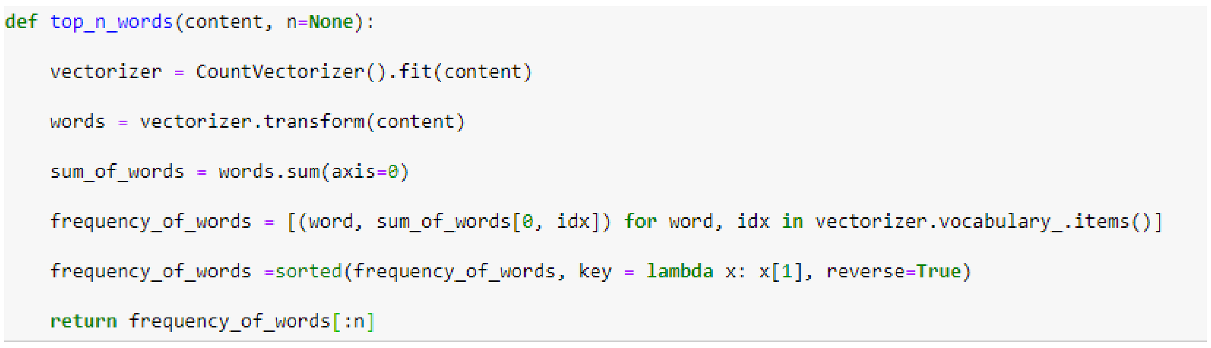


Figure 6.3 - Most used words in column paper_text after using stop words.

Figure 6.3 shows us that the words: the, as, in, to, is, and others, no longer exist and are removed from the dataset. With the removal of these words, the matrix will be smaller and the system will be faster for generating recommendations. Most used words now are learning, model, data, and algorithm. These words are very useful for the model.



```python
def top_n_words(content, n=None):

    vectorizer = CountVectorizer().fit(content)

    words = vectorizer.transform(content)

    sum_of_words = words.sum(axis=0)

    frequency_of_words = [(word, sum_of_words[0, idx]) for word, idx in vectorizer.vocabulary_.items()]

    frequency_of_words =sorted(frequency_of_words, key = lambda x: x[1], reverse=True)

    return frequency_of_words[:n]
```

Figure 6.4 - The function that is used for stop-words removal.

First, we define a function with def. With CountVectorizer we get the words model, where the text is clean, removing non-alphanumeric characters and stop words. With vectorizer.Transform we have a matrix, where specific texts are represented by rows, and words are represented by columns. All words and texts that exist in the content. Sum_of_words it’s just a vector that has the sum of each word that exists in the content. In the end, we make a list of tuples that contain the words.

The same method is applied for finding bigram and trigram words. A bigram is a two-word sequence of words, for example: of the, in the, to the, and the, is the, and others. Some examples of trigram words that we found in the corpus: in this paper, to, where is the, based on this paper we, the set of, and others.

6.5 TF-IDF

Stands for Team-Frequency – Inverse Document Frequency. It is using Machine Learning and Text Mining as a weighting factor for features. The weight increases as the word frequency in a documented increase.

This means that as the weight increases, the more that word occurs in the document.

As we can see not all words that appear in the corpus of documents have the same weight. But what if we have documents that describe cats. Then many of those documents will have in their content the word “cat” and it will have a high weight. In such cases, the TF-IDF comes into place.

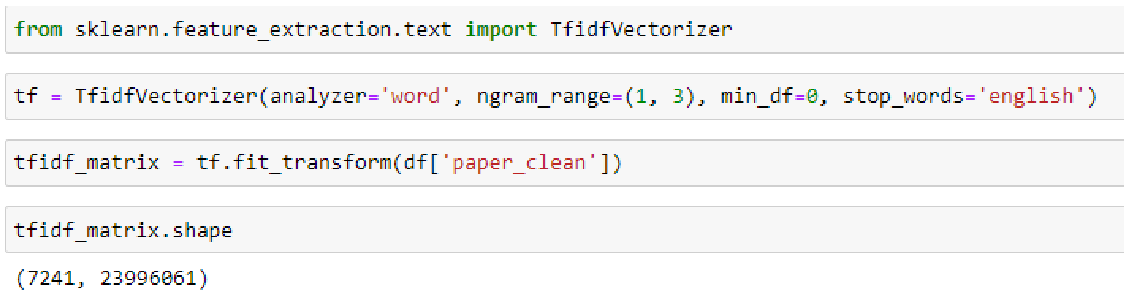
It assigns weight to each word according to this formula:

$$W_{i,j} = tf_{i,j} X \log log \left(\frac{N}{df_i} \right)$$

Figure 6.5 - Weighting formula.

(W I j) it describes the weight of the word (I) in the document (j), (d fi) describes how many documents contain the term (i), and the (N) describes the total number of documents.

From this, we understand that the weight of the word in a document is greater if it appears more often in that particular document, and less often in other documents.



```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')
```

```
tfidf_matrix = tf.fit_transform(df['paper_clean'])
```

```
tfidf_matrix.shape
```

```
(7241, 23996061)
```

Figure 6.6 - TF-IDF code in the system.

First, we import the TF-IDF Vectorizer from sklearn. Then we define a variable tf where we have the Vectorizer with ngram and the stop words. With tfidf_matrix we create our matrix based on the column paper_clean. This column has all the unnecessary words removed such as “stop-words”. At the end with .shape, we see that the dataset contains 7.241 papers which are described by 2.399.6061 words.

6.6 Cosine Similarity

It is a metric that is used for measuring the similarity between two or more items. Measures the cosine of an angle between two vectors that are represented in multi-dimensional space. With this method, we can measure the similarity of a document of any type. We can view the cosine similarity between two documents with the following formula:

$$cosine\ (x, y)\ =\ \frac{x \cdot y^T}{||x|| \bullet ||y||}$$

Figure 6.7 - Cosine Similarity for two documents.

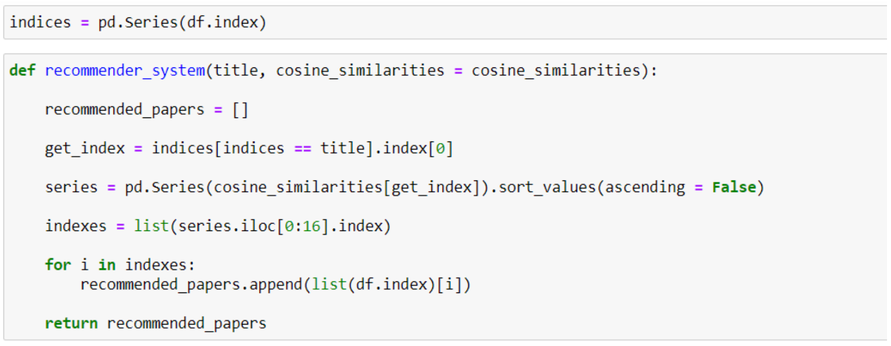
The cosine similarity takes a value between -1 and 1. The higher the value the more similar the documents are. If we get a value of -1 then there is no similarity between those two documents. In our case, we got values such as 0.7554, 0.8232, and others.

$$cosine\_similarity = linear\_kernel(tfidf\_matrix,\ tfidf\_matrix)$$

Figure 6.8 - Cosine Similarity code.

6.7 Making a Recommendation

After the creation of the Matrix, and then the cosine similarity now we have the scores of the documents and their similarities. It is time to find the index of that score and converted it to its title.

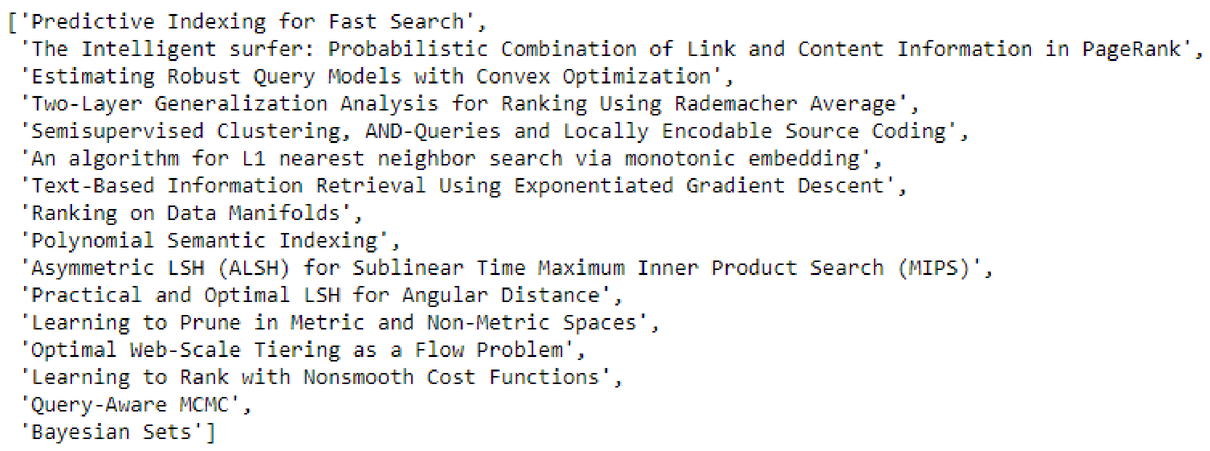


```
indices = pd.Series(df.index)
```

```
def recommender_system(title, cosine_similarities = cosine_similarities):

    recommended_papers = []

    get_index = indices[indices == title].index[0]

    series = pd.Series(cosine_similarities[get_index]).sort_values(ascending = False)

    indexes = list(series.iloc[0:16].index)

    for i in indexes:
        recommended_papers.append(list(df.index)[i])

    return recommended_papers
```

Figure 6.9 - Recommendation function.

As we mentioned above, first we get the indexes, then convert them to the title column, then we list the top sixteen papers that are similar to our particular search. With the following search: “recommender_system ('Predictive Indexing for Fast Search')” the results will be:



```
['Predictive Indexing for Fast Search',
 'The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank',
 'Estimating Robust Query Models with Convex Optimization',
 'Two-Layer Generalization Analysis for Ranking Using Rademacher Average',
 'Semisupervised Clustering, AND-Queries and Locally Encodable Source Coding',
 'An algorithm for L1 nearest neighbor search via monotonic embedding',
 'Text-Based Information Retrieval Using Exponentiated Gradient Descent',
 'Ranking on Data Manifolds',
 'Polynomial Semantic Indexing',
 'Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)',
 'Practical and Optimal LSH for Angular Distance',
 'Learning to Prune in Metric and Non-Metric Spaces',
 'Optimal Web-Scale Tiering as a Flow Problem',
 'Learning to Rank with Nonsmooth Cost Functions',
 'Query-Aware MCMC',
 'Bayesian Sets']
```

Figure 6.10 -Results of the system.

7 Chapter 7 - Evaluation of Recommender Systems

A proper evaluation is very crucial for the system, to understand the effectiveness of the recommendation algorithms. The evaluation of the recommender systems is multifaceted, meaning we cannot capture the whole design with just a single criterion. An incorrect evaluation might lead to underestimating or overestimating the accuracy of the algorithm.

Recommender systems can be evaluated in two ways: online and offline. In the online method, user reactions are measured and user participation is essential in this method. For example: in evaluating the news recommendation systems, the evaluator might look at the clicking rate in the articles that were recommended to the user. This method is known as A|B testing and measures the impact of the recommender system on the user. So, what is the most important thing here? Increasing the clicking rate of the user on the items.

As we mentioned above, using the online evaluation requires user participation, and this method is not feasible to be used in benchmarking and research. On the other hand, gaining access to a large dataset that has user conversion data is very challenging and one might not get full access, but only access into a particular part of the system.

Such access, limit the evaluation of the whole algorithm in different domains because evaluating an algorithm in many domains with different types of data is more useful and accurate. Due to this restriction, offline methods are created that use historical data. The offline method is mostly used when evaluating recommender systems for research and practice perspectives.

When using the offline methods, the accuracy of the algorithm cannot be verified fully, and we often cannot provide the full picture of the clicking rate of the algorithm. In this case, secondary measurements come into play. The most important part is the design of the evaluation system. It is crucial to design it carefully so it can be able to truly reflect the effectiveness of the algorithm. When trying to design an evaluation system we might face several issues:

- **Evaluation Goals:**the usage of accuracy metrics for evaluating a recommender system, doesn’t give a complete picture of the user experience. That’s why including secondary components such as novelty, trust, and serendipity is useful for the user experience. These metrics have both short-term and long-term impacts on the clicking rates.

- **Experimental Design:** even if we want to use the accuracy metrics, we have to take into consideration the design of the system, because it might lead to overestimation of the accuracy metric. For example: if we use the same set of ratings both for creating the model and late for evaluation of the model then the clicking rate will be highly overestimated.

- **Accuracy Metrics:** despite the secondary metrics that exist, the accuracy metric is the most important one when evaluating a recommender system. There are metrics such as mean absolute error (MAE) and mean squared error (MSE) that are used commonly in the evaluation process.

7.1 Types of Evaluations

The evaluation of systems is done in three types: user studies, online evaluations, and offline evaluations that include historical data. The user studies method and the online evaluation method, need user interaction to evaluate the system. They are not the same, and they do have differences in the technique of how the user is included in the evaluation system.

7.1.1 **User Studies**

This evaluation method includes test subjects that are used for interacting with the system and performing specific tasks. Feedback is collected from the users either before the interaction or after. The collection of this data in later stages is used to make inferences about the likes and dislikes of the user. For example we can ask a user to interact with the system in a particular site and give feedback about the items he gets recommended.

This approach later is used in judging the effectiveness of the algorithm. Another example: users can listen to different songs and in the end, they should provide ratings to each song.

When using this technique we can collect information about how the user interacts with the system. Different scenarios can be tested to make changes either in the user-interaction or into the algorithm or changing the user-interface. But there is one limitation. If the user is informed that he is tested, then his action and behavior might be affected. Including a large group of the user for testing purposes is quite hard. When asking users for participating in such evaluation systems, most of them will like to participate in their field, for example: in the song evaluation recommender, the testing participants will be mostly musicians. So not all users are from the general population and in the end, we cannot fully trust the results we get from this evaluation method.

7.1.2 **Online Evaluation**

This method also includes the user interactions but mostly the users are real users and the system is fully deployed online, meaning they are not there only for testing purposes. In this approach, users are using the system directly and are sampled randomly where various algorithms can be tested in different samples of users. As a metric typically conversion rate is used. This metric measures the frequency of which items are selected by the user. As we described above this method is known as the A|B testing and the idea is to compare two algorithms in the following way:

1. Having two groups of users (A and B).

2. Using one algorithm for group A and the other for group B and keeping the conditions similar in both groups.
3. In the end we compare the results (conversion rate) of the two groups.

Such an approach is good for testing the performance of the system in a long-term way. We are also able to not divide the users into groups, but by giving users both algorithms and keeping track of the results we get by seeing from which algorithm the result came.

The main disadvantage of the online evaluation is that it requires a large portion of users to be interacting with the system. Implementing this method into the startup phase is not required. Such tests are only available to the owners of the commercial entity, meaning scientists and other researchers don't have free access.

7.1.3 **Offline Evaluation**

This method works with historical data such as ratings. In some cases, in the data, a timestamp is included to show when the item was rated. Historical data we can find online, but the most knows in the Netflix Prize dataset. This dataset was published because of the online context that Netflix arranged and from then it still has been available online for testing algorithms. The advantage when using a historical data set is that it doesn't require access and interaction from the users.

The offline method is the most used technique to evaluate the recommender systems because many frameworks and evaluation measures have been developed for such cases. A disadvantage of this method is that it doesn't measure how the user will behave with this system. When using historical data, the problem is that the data might evolve (new data) and the current predictions that will be made by the algorithm might not include predictions of the future. Despite these disadvantages that exist, the offline method is still one of the most used when testing algorithms.

7.2 Goals of Evaluation Design

Some general goals that are very important in evaluation design are accuracy, diversity, robustness, scalability, and others. We will describe each of them in detail.

7.2.1 **Accuracy**

It is one of the most fundamental measures when evaluating recommender systems. This method is formed by two main components:

- **Designing the Accuracy Evaluation:**this component describes to us that we cannot use the same rating matrix for both pieces of training the model and accuracy evaluation. If we do so then we will overestimate the results. So it is very important to use a different set of entries for training and accuracy.

- **Accuracy Metrics:** This is used to evaluate either the accuracy of the ratings from specific users or the accuracy of top recommended items predicted by the system. The technique uses mean squared error and root mean squared error.

$$MSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^{2}}{|E|}}$$

Figure 7.1- Mean squared error.

(u) is the user, (j) the item, (E) is the set of entries.

$$RMSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^{2}}{|E|}}$$

Figure 7.2- Root mean squared error.

Some of the measures are designed to maximize the profit for the merchant. The main problem with this method is that it cannot measure the real effectiveness of the recommender system. For example, a recommendation might be accurate, but in the real world, the user may buy that item anyway.

7.2.2 **Coverage**

With coverage, we understand the percent of items that were used in the training data that the algorithm was able to the recommender. That means the recommenders not able to include all the items in the prediction accuracy. This limitation of the recommender system exists because the rating matrix is sparse. In practical scenarios, the system often has 100% coverage. There are two known ways of coverage: user-space coverage and item-space coverage.

The user-space coverage is used to measure the fraction of users in which k ratings may be predicted. We should give a value to k, and if the list of recommended items is lower than that value then the recommendation is not meaningful. For example, for a user that has rated just two movies, it’s hard to mutual that user with other users.

Item-space coverage measures the fraction of the items that the recommender can predict. For example: if we have a system that has lower item coverage then this limits the recommender to recommended items.

7.2.3 **Confidence**

We can define this as how much the system trusts its recommendations. As we said in the previous sections that when a large amount of data are available then the collaborative system can make better recommendations, similarly the confidence of the system also gets higher. For example: if the recommender system recommends to the user two movies, one with high confidence, and the other with low confidence then the user might be affected and will add the first movie into the watching list, and for the second movie he will do more research or will ignore it.

7.2.4 **Trust**

While the confidence part had to do with the system trust in its ratings, with trust we refer to how much the user trusts the system that is recommending items to him. As a start, the system should recommend to the user items that the user is already familiar with. This way, the user may not benefit from the items but the system will benefit from the user because it will gain his trust. Another method of getting trust is by describing to the user how you recommended the item to him. If we do not restrict ourselves to just one method, then we can try different methods and grow the credibility of the system. But how can we check the trust of the system? One method is by asking users for their feedback on how much they trust the system. Another method is by checking users. After the user has used the system one time, and it returns then we can assume that he has trust in the system. Measuring trust in the offline experiment is unclear because trust is built through the interaction of the users with the system.

7.2.5 **Novelty**

Represents those items that the user hasn’t seen. In those applications that require a novel recommendation, we can filter all the items that the user has rated and we will know which items he has not seen. But in some cases, the user has seen the item but didn’t give feedback and we may recommend to him some items that he has already seen in the past. Another method is by asking the user whether he has seen a particular item. We can gain some novelty in offline experiments. By splitting the data into the specific point of times, hide all purchases that the user has made in a specific time, then test what items the system will recommend.

To implement this method we need to carefully model the hiding processes of items. When using novelty we need to take into consideration the accuracy, because the user will get irrelevant items recommender and they may be worthless. Novelty is better to be considered into relevant items.

7.2.6 **Serendipity**

With serendipity, we measure how surprising is the recommendation system. For example, a user has watched movies where the actor was Brad Pitt. Based on this information the recommender system will know that this user may be a fan of this actor and he will recommend to the user a movie where Brad Pit is acting. But random recommendations are more surprising so we need to make a balance between accuracy and serendipity. We can think of serendipity as new information from the user. For example a new movie appears (The Joker), and if the user rates this movie as 4 or 5 then he likes it. Based on this feedback we will recommend to him more movies when Joaquin Phoenix is acting.

To evaluate the serendipity of a recommender system we use user studies. Asking the user which of their recommended items are unexpected to them. Then we can see if the user liked these recommendations, which will make the system more successful and serendipitous. But we need to take into consideration that too much serendipity is bad for the system because in the starting phase the user may like the new unsuspected movies but later he may discover that the recommendation system is inappropriate and will stop using the system, that's why checking the score of the serendipity is suggested.

7.2.7 **Diversity**

It's just the opposite of similarity. We have seen that suggesting similar items to the user may not be always useful. Example: suggesting a vacation place to a user. If you suggest five hotels in the same location it may not be quite useful versus suggesting five hotels in five different locations. The user will have more possibilities in searching about the locations and requesting more details.

To measure the diversity we use the item to item similarity that is based on the content of the item. This measure is not the same as the similarity measure that is used to recommend items to the user.

7.2.8 **Utility**

It is very important for e-commerce websites. Many of these websites use a recommender system to improve their revenue (selling more). Based on this, we can judge a recommender by the revenue that they generated for a particular website. For these recommenders, measuring the utility is more important than measuring the accuracy of the recommender. The utility can be measured either by the recommendation system or by the owner of the system. In some cases where we have items that get rates, we can use the ratings to measure the utility. For example: if a user rates a movie as five stars then we assume that this movie has excellent utility versus a movie that is rated with four stars.

We can assign both positive and negative utilities to a particular recommender system. For example: if some items that are recommended to the user are offensive then we rate that system with negative utility.

7.2.9 **Risk**

Risk is also available in the recommendation system. For example: when recommending stocks to users, the user may like the stocks to be risk-averse and will prefer stocks that have a low risk. On other hand, some users may be risk-takers and want stocks that have higher risks so they can gain more profit. To evaluate risk systems we consider not just the utility but also the utility variance.

7.2.10 **Robustness**

With robustness, we understand how much the recommendation system is stable against fake information that is added into the system to influence the recommendations. Many people rely on the recommendation system for their decision, and influencing the system to change ratings may be very profitable for certain people. Example: a website that recommends hotel rooms (booking.com).

An owner of some hotels may want to boost their rating for their hotel and they will inject fake users to rate their competitors negatively.

These attempts to influence the system are known as “attacks”. These attacks occur when coordinated users inject fake information into the system to benefit themselves. When evaluating these systems, we need to write a complete description of the attack protocol, because the system is very sensitive to these protocols.

Creating a system that is immune to these attacks is nearly impossible. We need to simulate attacks to see how our system will be affected. But it is hard doing an attack on a real system that is online. It’s more beneficial to analyze the data that the online system gathered and identity attacks that have occurred in that system.

Another type of robustness is the stability of the system against a large number of requests. This is very important for system administrators, who want to avoid system malfunction. This depends a lot on the infrastructure such as database, servers, and the scalability of the system.

7.2.11 **Privacy**

In some systems, such as collaborative filtering, many users willingly disclose their preferences over items, to get better recommendations. But we need to consider that the privacy of users is very important and their data should stay private, to avoid third-parties from using the recommender system to get their data and gain benefits. For example, a user is interested in the field of Artificial Intelligence and has bought the book “Life 3.0: Being Human in the Age of Artificial Intelligence”. Now, when the spouse of that user will use the system, to buy a present, she may get a message such as “people who bought this book, also got interested in”. These kinds of scenarios reveal sensitive information about the user.

To avoid such scenarios, it is appropriate to define different levels of privacy. By focusing a lot on privacy, we may affect the accuracy of the recommender system. We need to do such modifications about privacy that the system with or without that modification will be able to recommender the item with the same accuracy.

7.2.12 **Adaptivity**

Nowadays recommender system operates in those settings where trends are changing rapidly, and there are more and more items available in the market. Such examples are:

news recommender systems, where a big event might occur (volcano explosion) and users will be interested in the phonemes of volcanos, and the system should be able to dig in the past and find articles that are related to volcanos and suggest those articles to the users.

The evaluation of adaptivity is done with offline techniques, simply by analyzing how much information does the system needs before we recommending an item to the user.

Another type of adaptivity can be the rate of the system for adapting to the user's preferences, or the changes of preferences in his profile. For example: when a user rates a certain item, he will also expect that the system will change its recommendations. If the system doesn't change, then the user will think that the rate he did, didn't affect the system and it is useless, and this may impact the user to not provide ratings in the future.

7.2.13 **Scalability**

Recommender systems are built to help and guide users through large collections of items. When building such systems, it is very important to scale them. An algorithm when trying to find items for the users through this dataset will trade accuracy and coverage over speed.

The growth of the dataset has an impact on the speed of the algorithms. Many algorithms are slowed down or are required to get more resources (power and memory).

To measure scalability, we use growing data sets, and we view how the data will affect the speed and system resources when trying to finish the task. The important part is to view how the accuracy will be affected and what kind of other types of compromises the scalability will dictate. The speed of the recommender system is also important and we can measure that by viewing how many recommendations the system can provide per second.

8 Chapter 8 – Evaluating the System

Every recommender system needs a proper design of the evaluation system. The evaluation method is done either with online methods or offline methods. Since the online evaluation requires user's activity and participation, we used the offline methods that are often used in benchmarking and research. To have a clear picture of the design of the recommender systems, the evaluation cannot be based only on single criteria. Taking into consideration this, we designed the evaluation process with different techniques such as: precision and recall, novelty, serendipity and diversity.

8.1 Precision-Based Metrics

8.1.1 Precision and Recall

Evaluation plays a huge role in estimating the performance of a recommender system. To evaluate our recommender systems we have used the method of Precision and Recall. These evaluation metrics are often used by Data Scientists to evaluate the accuracy of their system. With this method, we can measure the classification model's accuracy.

- Precision is the measure of how many observations our model correctly predicted over the amount of correct and incorrect predictions.

  Precision is calculated with: *Precision = TP / (TP + FP).* TP means True Positive and FP means False Positive.

  For example: if we measure cars versus bikes, precision will measure the number of correctly predicted cars divided by the cars that are *correctly* labeled as cars and bikes *incorrectly* labeled as cars.
- Recall is the measure of how many observations our model correctly predicted over the total amount of observations.

Recall is calculated with: *Recall = TP / (TP + FN).* FN means false negative.

For example: in the example of cars and bikes, recall measures the number of cars labeled correctly divided by the total amount of cars present(Santos, 2020).

8.1.2 Implementation

The Precision and Recall method mentioned above is the classical method. Due to a lack of data in our dataset we were not able to implement the classical method of precision and recall. By believing that every recommender system should be tested in accuracy and by taking into consideration our deep desire to test the accuracy of our recommender systems, we went for a logical method of precision and recall.

In our system, we measure the similarity between papers. In this example, we will measure the similarity between the first paper versus the second paper, third paper, and fourth paper. This list of papers is based on the results we got from the recommender system. In the end, we will see if these papers have similar words and which paper has the most similar words with the first paper. If both papers have common words then it is likely they can be very similar. So we would compare them based on their common words.

The process starts with exporting the *paper_clean* column from the recommender system into a CSV format. Then we split these four papers and put them into empty text documents.

Then we declare an empty string and with the help of a *for* loop, we stripe the line and convert the text in lowercase. The purpose of this conversion in lowercase is to be sure all the text is in lowercase so we can compare it. This process is done with all four papers.

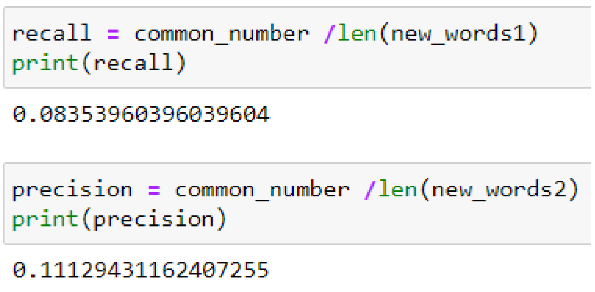
In the second phase we remove the stop words such as (to, that, and, this, etc.) After we clean the text, in the third phase we search for common words between the two papers.

Based on the results we got, paper one/two have 135 common words, paper one/three have 134, and paper one/four have 104 common words.

In the end, we calculate the accuracy metrics with precision and recall..

$$Recall = \frac{Common\ words}{Paper\ 1}$$

$$Precision = \frac{Common\ words}{Paper\ 2}$$



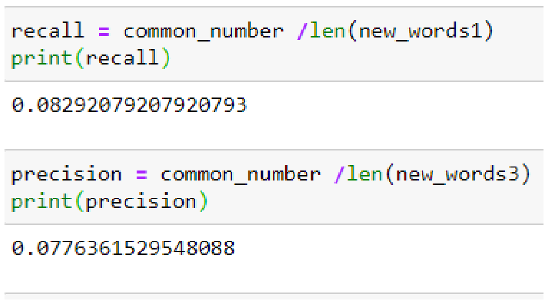
```
recall = common_number /len(new_words1)
print(recall)
```

```
0.08353960396039604
```

```
precision = common_number /len(new_words2)
print(precision)
```

```
0.11129431162407255
```

Figure 8.1–Paper 1 vs Paper 2



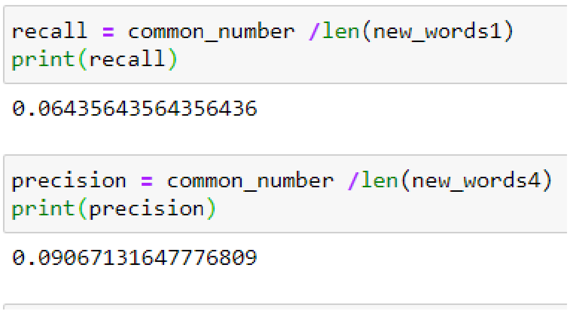
```
recall = common_number /len(new_words1)
print(recall)
```

```
0.08292079207920793
```

```
precision = common_number /len(new_words3)
print(precision)
```

```
0.0776361529548088
```

Figure 8.2 – Paper 1 vs Paper 3



```
recall = common_number /len(new_words1)
print(recall)
```

```
0.06435643564356436
```

```
precision = common_number /len(new_words4)
print(precision)
```

```
0.09067131647776809
```

Figure 8.3 – Paper 1 vs Paper 4

In conclusion, we see that in figures 8.1, 8.2, and 8.3 we have recall results and precision results. Based on those results, we conclude that the paper with the most similar text to paper one is paper two with a recall score of (0.08353), then paper three (0.08292) and paper four(0.06435).

8.2 Other Metrics

By considering that different applications have different needs, where many characteristics are added to the recommender system, the evaluation method goes beyond accuracy and precision. It is important to include alternative metrics and integrate them into the evaluation process.

8.2.1 Novelty

The novelty method is based on the like hood of the user receiving items that they are not aware of and that they have not seen before. This is very important, because the user may discover important items that they did not know previously. Taking into consideration this, we concluded that in content-based recommender systems the novelty is low because the system recommends items that tend to be somewhat obvious. The best way to measure novelty is through online experiments, where the user will be asked the question of whether they were aware of that item previously.

8.2.2 Serendipity

Serendipity has to do with measuring the level of surprise in recommended items. Is the recommender able to recommender unsuspected items? Serendipity has a stronger condition versus novelty. Serendipity in content-based recommender systems has a low score because their systems rely on the context of the items that previously were rated by the user.

8.2.3 Diversity

The notion of diversity stands for those types of recommender systems that suggest diverse items to users. The recommended items should be as much diversity as possible. For example, if a user receives a recommendation of the top three movies and all three movies are of the same genre and have similar actors, the user may dislike the top choice and there is a great chance that he will skip all of them. But if we present different types of movies, that increases the chance of the user selecting one of them.

Knowing that content-based recommender systems provide obvious recommendations because of their use of the content, may reduce the diversity of the recommended items which is undesirable by the user.

9 Chapter 9 - Conclusion

9.1 In Summary

In this thesis, we tackled the problem of recommending computer science publications to users. We described the importance of feature selection and its big impact and role within the processes of the recommendation system. Although it is possible to use any kind of representation data, the common approach is to use data that are not available in a variety of domains. There are cases when items have multiple fields that describe the various aspect of the item. For example, in our recommender system, we had seven fields for a paper, but we only used three of them. The purpose of feature selection is to ensure that only important and informative words are in the vector-space representation. In some cases, very good recommenders suggest that a size cut-off is needed to remove some noisy words that can result in overfitting. This is often used in small datasets where the number of items is small and there is a great tendency for the model to overfit. In feature informativeness, we have two parts: feature selection and feature weighting. The feature selection part corresponds with the removal of words that was implemented in the recommender system with the method *"stop_removal_words",* and feature weighting aligns with giving greater importance to words by using the inverse document frequency. These two types of feature selection belong to the unsupervised method, where the supervised method has to do with user rankings.

In Chapter 3 we described that Collaborative systems do not use item attributes when computing predictions. This is a huge downfall because if Chris likes Spider-Man then there is a huge chance he may like a movie from the same genre, such as Justice League, Black Panther, and others. In such cases, the content-based systemdoesn't need user ratings to make a recommendation, instead, it recommends items by relaying at the own user rating and the attributes of the item.

By creating a content-based recommender system, we concluded that these systems are depended on two sources of data:

1. Description of items: for example, a text description for a particular item.

2. User profile: this source is generated by the user’s feedback about particular items, either by rating the item or by the user’s action.

In this scenario, we create the similarity between two publications based on the text attribute, put them into a vector-space representation, and by using inverse document frequency we weigh the documents.

To find similarities between the documents we used a similarity function. Cosine similarity is a great function and is often used in the text-domain. The usefulness underlines in making predictions for particular items (in this case documents), where we don’t have a lot of information about the user's preferences.

Our results show that the content-based approach has user independence because the recommender can suggest items by analyzing only a single user profile and the items. It has transparency because the items are recommender from a feature-level basis. And the most important one is that has no cold start problem because new items can be recommended to the users without other users rating the item.

9.2 Advance Topics

9.2.1 Evolution

“*In the last fifty years, science has advanced more than in the two thousand previous years and given mankind greater powers over the forces of nature than the ancients ascribed to their gods.*”– John Boyd Orr(Aggarwal, 2016)

Recommendation systems are evolving in multiple directions. There are three phases of the evolution of recommender systems:

- Phase 1: General Recommendations
- Phase 2: Personalized Recommendations
- Phase 3: Futuristic Recommendations

In the first phase, we have a system such as collaborative filtering, user-based recommendation systems, and items-based systems. These are the earlier generation of recommender systems.

In phase two, we have an explosion in information, where people started using the Web more often and leaving digital footprints such as clicks, browser history, search patterns, and others. Companies started looking to what kind of items the user likes, and which features of the item are making the user look for it. In this phase, the so-called content-based recommender systems were created.

Currently, we are in the earlier stages of the third phase. In this phase, ubiquitous recommender systems are created. Such systems can recommend items to the user in real-time, based on the location of the user, their mood, sleep cycle, and others.

The main purpose in this field should be to create a sophisticated recommender system that will be able to predict the user's next move and make suggestions without him asking for it.

9.2.2 Group Recommender Systems

These types of recommenders are very interesting and challenging because of their objectives and goal. In this type of recommender system, we have to deal with a group of users, not a single user. Such cases and needs appear in many areas, for example: when watching a movie with a group, playing music in a fitness center, travel recommendation to a group of tourists. The earliest systems, take into consideration the user's individual needs and aggregate them into a group of preferences. But the need in this field is to create recommender systems that are not just based on the sum of user's individual preferences, but also considering user's interaction.

9.2.3 Privacy in Recommender Systems

Many of the recommender systems are based heavily on user feedback, whether is implicit or explicit. The problem underlines in the feedback part because they contain significant information about the user's interest, and such information might reveal their political opinion, sexual orientation, and some personal preferences. Such privacy concerns are significant and in recent years the privacy topic has exploded in a variety of many data mining areas.

Bibliography

Achakulvisut, T., Acuna, E., Ruangrong, T., & Kording, K. (2016). Science Concierge: A fast content-based recommendation system for scientific publications. *PLoS ONE*, 11(7): e0158423.

Aggarwal, C. C. (2016). *Recommender Systems.* Yorktown Heights, NY, USA: IBM T.J. Watson Research Center.

Analytics, I. D. (n.d.). *Data Analytics*. Retrieved from IBM: https://www.ibm.com/analytics/data-science

Billius, D., & Michael, P. J. (2000). User Modeling for Adaptive News Access. *Kluwer Academic Publishers*, 147-180.

Breese, J. S. (1998). *Proceedings of the Fourteenth Conference on Uncertanity in Artificial Intelligence.* Madison, WI. .

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *https://doi.org/10.1023/A:1021240730564*, User Model User-Adap Inter 12, 331–370.

Castillo, D. S. (November 6, 2007). Hybrid Content-Based Collaborative-Filtering Music Recommendations. 6-60.

Cioffi, R., Travaglioni, M., Piscitelli, G., Petrillo, A., & De Felice, F. (2020). Artificial Intelligence and Machine Learning Applications in Smart Production: Progress, Trends, and Directions. *MDPI*, 1-26.

Das, A. D. (2007). *Google news personalization: scalable online collaborative filtering.* New York.

Devlin, J., Chang, M.-W., Lee, K., & Tautanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*.

Dron, J. (2009). *Self-Organization in Social Software Learning.*

Geeksforgeeks. (2019). Decision Tree Introduction with example. Повратено од https://www.geeksforgeeks.org/decision-tree-introduction-example/

Gipp, B., Beel, J., & Hentschel, C. (2009). Scienstein: A Research Paper Recommender System. *Conference Paper* (стр. 1-7, PaperID: 213 1). Virudhunagar, India: IEEE.

Google. (2018). Повратено од GitHub: https://github.com/google-research/bert

Gorakala, K. S. (2016). *Building Recommendation Engines.* Birmingham, B3 2PB, UK.: Packt Publishing Ltd.

GroupLens. (1997). MovieLens. *Wikipedia*.

Hamner, B. (2017, 12 05). *NIPS Papers*. Retrieved from Kaggle: https://www.kaggle.com

Horev, R. (2018). BERT Explained: State of the art language model for NLP. *Towards Data Science*.

Jannach, D., Zanker, M., Felfering, A., & Friedrich, G. (2011). *Recommender Systems.* 32 Avenue of the Americas, New York, NY 10013-2473, USA: Cambridge University Press.

Jannach, D., Zanker, M., Ge, M., & Groning, M. (2012). Recommender Systems in Computer Science and Information Systems – A Landscape of Research. *EC-Web 2012* (стр. 76–87). Verlag, Berlin, Heidelberg: Springer.

Lenhart, P., & Herzog, D. (September 16, 2016, Boston, MA, USA). Combining Content-based and Collaborative Filtering for. (стр. 1-8). Boston, MA, USA: CBRecSys@RecSys.

Manning, D. (2008). *Introduction to information retrieval.* Cambridge.

Mikilov, T., & Le, V. (2014). Distributed Representations of Sentences and Documents. *arXiv.org*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv.org*.

Mohanty, A. (2020). Doc2Vec: An Extension To The Word2Vec. *Medium*, 1-2.

Nandi, R. R., Zemam, A. M., Muntasir, A. T., Summit, H. S., Surov, T., & Rahman, J. U.-r. (2018). Bangla News Recommender system using doc2vec. *ResearchGate*, 1-8.

Oladapo, F. B. (n.d.). *A RESEARCH PROPOSAL ON PAPER RECOMMENDATION SYSTEMS.* Retrieved from www.academia.edu: https://www.academia.edu/4041704/Content_Based_Recommendation_Systems

Patel, H. (2013). Seminar on Recommender systems using Hadoop. *U-News*, 1-3.

Pazzani, M., Muramatsu, J., & Billsus, D. (1996). *Syskill & Webert: Identifying Interesting Web Sites.* AAAI/IAAI, Vol. 1.

Philip, S., (PHD), S. P., & E.P, M. (2014). A Paper Recommender System Based on the Past Ratings of a User. *International Journal of Advanced Computer Technology (IJACT)*, Volume 3 - Number 6.

Philip, S., Shola, P., & John, O. A. (2014). Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library. *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 5, No. 10.

Puleston, E. (2012, 08 21). User Ratings.

ResearchGate. (2008). A linear classifier separating two classes. Retrieved from https://www.researchgate.net/figure/A-linear-classifier-separating-two-classes-of-points-squares-and-circles-in-two_fig1_23442384

ResearchGate, S. F. (2020, 11 26). Characteristics of the books analyzed. Retrieved from https://www.researchgate.net/figure/Characteristics-of-the-books-analyzed-The-length-of-each-book-L-is-measured-in-millions_fig1_264425089

Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender Systems Handbook - Second Edition.* New York, US: Springer.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender Systems Handbook.* Springer Science+Business Media, LLC, 233 Spring Street, New York,: Springer .

Santos, M. (2020, May). *Explaining Precision vs. Recall to Everyone*. Повратено од https://towardsdatascience.com/.

Sarwar, B. (2001). Item-based collaborative filtering recommendation algorithms. *ACM*.

Sejnowski, T. (2015). *Neural Information Processing Systems*. Retrieved from NIPS: https://nips.cc/

Singla, L. (2019). Why Building a Recommendation Engine is a Good Strategy for Your eCommerce Business? *Netsolutions*, 1-3.

Stecanella, B. (2019). What is TF-IDF? *MonkeyLearn*, 1-3.

Suresh, G. K. (2015). *Building a Recommendation with R.* 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing.

T, A., DE, A., & T, R. (2016). A Fast Content-Based Recommendation System for Scientific Publications. *PLoS ONE*, 11(7): e0158423.

Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. *Elsevier*, 1-9.

Yang, Y., Adeniyi, D., & Wei, Z. (2016). Personalised news filtering and recommendation system using Chi-square statistics-based K-nearest neighbour. 1-20.