VIETNAMESE - GERMAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Frankfurt University Of Applied Science
Faculty 2: Computer Science and Engineering

COMPARATIVE ANALYSIS OF PERSONALIZED RECOMMENDER SYSTEMS FOR MASTER'S PROGRAMS: EXPLORING DIFFERENT APPROACHES

Full name: Vinh Nguyen Phuoc Bao Minh

Matriculation number: 1403541
Supervisor: Dr. Tran Hong Ngoc
Co-supervisor: Dr. Dinh Hai Dung

BACHELOR THESIS

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering in study program Computer Science, Vietnamese - German University, 2024

Binh Duong, Vietnam

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

In addition, I certify that this submission contains no material previously submitted for award of any qualification at any other university or institution, unless approved for a joint-award with another institution, and acknowledge that no part of this work will, in the future, be used in a submission in my name, for any other qualification in any university or other tertiary institution without the prior approval of the University, and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of any published works contained within this thesis resides with the copyright holder(s) of those works.

I give permission for the digital version of my research submission to be made available on the web, via the University's digital research repository, unless permission has been granted by the University to restrict access for a period of time.

Vinh Nguyen Phuoc Bao Minh
Computer Science Student
Vietnamese German University
Date: November 26, 2024

Signature: ____________________

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Tran Hong Ngoc, for her invaluable guidance, insights, and expertise throughout the course of this research. Her constant constructive feedback have been instrumental in shaping this work, and I am deeply appreciative of the support.

I am profoundly thankful to my family for their unwavering belief in my abilities and for providing me with constant motivation and emotional support. Their patience, understanding and **Mandu** dumplings during the many late nights and challenging moments have been a pillar of strength.

To my close friends, thank you for your encouragement and for being a source of inspiration and camaraderie. Your belief in me has meant more than words can express.

I am also grateful to my colleagues for their insightful inceptions of this research.

Finally, I extend my appreciation to all who have contributed, directly or indirectly, to the completion of this thesis. The implementation and supplementary resources for this research are available at my GitHub repository [2].

Abstract

This thesis investigates the development and evaluation of recommender systems (RecSys) in an academic domain, addressing common challenges such as data sparsity, and the absence of user preferences and ratings in real-world datasets. An extensive and in-depth literature overview of well-known RecSys is included to provide context and highlight existing advancements in the field.

The study explores both Content-Based Filtering (CBF) and Collaborative Filtering (CF) techniques, integrating natural language processing, NLP-based approaches and synthetic data generation to create accurate, robust and personalized recommendations.

The research initially constructs a synthetic user preference dataset using OpenAI's ChatGPT and extracting program data from a university portal. This approach compensates for the lack of explicit user preference data commonly found in real-world databases. The study determines that, to some extent, synthetic data can replicate real user input, providing a feasible solution for training and evaluating recommendation systems in the absence of real-world datasets.

The NLP-based CBF algorithm serves as the foundation for the system, utilizing `Term Frequency-Inverse Document Frequency (TF-IDF)` weightings and `Word2Vec` word embeddings to extract meaningful features from program descriptions and synthetic user preferences. Weighted sentence embeddings are then calculated by combining `Word2Vec` vectors with `TF-IDF` weights, to compute cosine similarity scores. These scores serve as the basis for personalized recommendations for users. These recommendations demonstrate that the NLP-based CBF approach effectively aligns program attributes with user preferences.

In addition, the study evaluates CF techniques, including User-Based CF, Item-Based CF, Singular Value Decomposition (SVD), and Non-Negative Matrix Factorization (NMF). Experiments under varying data sparsity levels suggest that SVD performs best in the academic recommendation scenarios due to its ability to handle large datasets efficiently. Also, the experiments show that increasing data sparsity negatively impacts recommendation accuracy, as evidenced by higher Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics across all methods.

This research introduces novel contributions, including the use of ChatGPT for synthetic dataset generation and the application of NLP-based CBF for academic program recommendations. While the findings highlight the strengths and limitations of various recommendation algorithms, the study also underscores the need for further testing on real-world datasets and exploring hybrid systems to overcome current challenges.

Keywords
Recommender Systems,
Collaborative Filtering,
Content-Based Filtering,
Word Embeddings,
Data Sparsity

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Since its first conception in 1979, Recommender System (RecSys) has been developed and researched extensively. RecSys has initially been introduced as a library application called Grundy in which it is used for book recommendation [40]. However, not until 1989, at the beginning of the Internet era, RecSys emerged as one of the powerhouse concepts in the area of Artificial Intelligence (AI) [39]. Then, in the 1990s, with the newly introduced concept of Collaborative Filtering for an experimental mail system named Tapestry [12], more and more RecSys development techniques have been proposed and implemented.

More recently, with the rise of big data and popularity of interest in machine learning, the use for commercialization on RecSys has been tremendous. For example, in the entertainment domain, Netflix deploys its recommender algorithms to analyze viewing patterns to suggest personalized contents to its users [52, 13] while Spotify uses playing history, audio signals and lyrics to suggest personal songs and playlists for its users [4, 53].

While RecSys applies plentiful in entertainment but other application domains have been left under-explored. In this thesis, we shift our focus to the education domain.

1.1 Motivation

On a more personal note, in my search for a Master's degree in Europe, I encountered an overwhelming number of websites. Germany, on its own, offers a vast array of educational options, with around 22,000 different courses, including over 2,000 related to computer science. I found navigating through specifically each and every university in more countries such as the Netherlands, Switzerland, and others to be an impossible task. Additionally, during my internship working with a shipping company, I worked firsthand on a commercial recommender system. In a nutshell, the company analyzed past shipping behavior and used RecSys to suggest products and services that would appeal most to the customers.

In general, the process of manually going through hundreds (if not thousands) of programs can be *time-consuming* and potentially results in *bad decision-making* and *overlooked opportunities*. These things combined, sparks the idea of exploring how such recommendation techniques could be used for higher education.

1.2 Problem Statement

Despite the advancements in recommender systems (RecSys) and their successful implementation in various industries, to the best of my knowledge, their application in higher education remains underexplored. Traditional RecSys often rely on extensive user interaction data, which is not always available or feasible to collect due to the increasing privacy concerns or non-trivial technical problems. Although researchers have been working to cope with these issues and have devised solutions that, to some extent, mitigate these issues; there is still a need for alternative approaches that can provide personalized recommendations with the minimal user input.

1.3 Research Questions and Objectives

Building upon the challenges identified in our aforementioned problem statement, this thesis seeks to explore innovative solutions for *personalized Master's program recommendations in the absence of real user interaction data.*

Main research question: How can various recommender system methodologies be effectively designed, implemented, and evaluated under the absence of user interaction data?

Specifically, the following objectives encapsulate the main focus of this thesis:

Objective 1. Literature review of Recommender Systems. We conduct a comprehensive literature review on different state-of-the-art recommender system methodologies, both in theory (i.e., in academia) and in practice (i.e., in applications), of different application domains.

Objective 2. Create a database of user's study preferences. We create, to the best of our knowledge, the first synthetic database of users' Master's study preferences. This database simulates real-world prospective students' interests and requirements for Master's programs.

Objective 3. Evaluation of our database and existing recommendation methodologies. We first evaluate the synthetic user database from Objective 2, how realistic it is to replicate real user data. Then, we apply the database to some notable methodologies from Objective 1 and perform an evaluation on different metrics (namely, relevance and accuracy).

1.4 Contributions

In this bachelor thesis, we create, using machine learning techniques, the first synthetic database of user' Master's study preferences.

Secondly, we conduct a classification survey between AI-generated and human-written statements within the context of the preferences of prospective students. This survey is designed to evaluate whether replicating the user with synthetic data is practical.

Thirdly, we implement a comparative evaluation of different RecSys methodologies within the context of the Master program recommendations. Each RecSys is evaluated based on multiple metrics, including accuracy, diversity, and novelty.

These findings can offer perspectives on which methodologies are more effective and can help universities and educational platforms understand which recommender system methodologies might best serve their prospective students, improving the effectiveness of the program recommendation service.

In addition, the methodologies and evaluation framework can be adapted to other domains where user data is scarce or where synthetic data can be utilized, extending the impact of the research beyond education.

Organization. This thesis is organized as follows: Chapter 2 refers to related work on recommender systems and existing examples in this system area. Subsequently, Chapter 3 provides an overview of well-known recommender systems methodologies introducing their fundamental concepts, advantages, and limitations. Next, in Chapter 4, we mention how synthetic user data are created and used to replicate real user inputs. We then present the evaluation of the synthetic user preference in Chapter 5, and the comparative analysis in Chapter 6 based on the evaluation metrics, including the measures of accuracy and relevance. Finally, we provide a general discussion and conclude this work in Chapter 7.

Chapter 2

Literature Review

In this chapter, we summarize existing literature relevant to the development and evaluation of recommender systems related to the thesis topic. In particular, we start with an overview of the *Recommender Systems* in Section 2.1: both theories and practices. Then, in Section 2.2, we shift the topic of focus to *Recommender Systems in Education*. Subsequently, we explore how the integration of *Natural Language Processing (NLP) into Recommender Systems* (in Section 2.3) and the use of *synthetic user data* (in Section 2.4) can overcome these challenges.

2.1 Recommender Systems

By definition, a recommender system, also known as a recommendation system, is an instance of information filtering system that provides items' suggestions to particular classes of user may be interested in [39, 43].

Table 2.1 displays some examples of the domain where recommender systems are currently used, along with example of some leading companies in such domains and type of information their systems recommend. For example, in the entertainment field, Spotify has recently released a new "Smart Shuffle" feature that allows the systems to *recommend* new songs that the user may be interested in [28], while YouTube and Netflix focus on having their systems recommending videos or movies.

Domain	**Examples**	**Type of Information**
E-commerce	Amazon, Alibaba	Products
Entertainment	Spotify, YouTube, Netflix	Music, Videos, Movies
Social Networking	Facebook, Instagram	Posts, Friends, Groups
Education	(See Table 2.2)	
News	Google News	News Articles
Travel	Airbnb, Booking.com	Accommodations, Destinations

Table 2.1: Examples of Recommender Systems

As far as research is concerned, Pioneer MovieLens [14], developed by the GroupLens Research Group, is a research platform that allows users to receive personalized movie recommendations based on their past ratings and preferences. This line of work has been widely used in academic research to study recommendation algorithms and user behaviors due to its publicly available datasets [38].

Common techniques used in recommender systems includes Content-based Filtering, [4], Collaborative Filtering, [10], Knowledge-based Filtering [30], as well as the Hybrid methodologies [54]. We will introduce and discuss these techniques in the subsequence chapter.

2.2 Recommender Systems in Education

Examples	**Type of Information**
Coursera, EdX, Udemy	Courses
Duolingo	Feedback and Practice Suggestions
Moodle	Resource Recommendations

Table 2.2: Examples of Recommender Systems in Education

In education, RecSys are typically applied in recommending courses to the users (e.g., students). RecSys offers a solution by finding contents relevant to their goals, skill levels, and learning pace, which enhances engagement and fosters a more individualized learning experience.

More than ten years ago, the Big Three of Massive Open Online Course (MOOC) platforms - Coursera, edX, and Udemy - revolutionized e-learning by making quality education accessible to global audience [8, 33]. Since then, with the improvement made in RecSys, these platforms have steadily enhanced the learning experience by personalizing course suggestions based on students' interest, skill developments and learning objectives.

In Learning Management Systems (LMS) such as Moodle, RecSys are used to recommend resources and activities that complement course content, helping students navigate through a wide range of educational materials [48]. Traditional LMS platforms, however, typically only provide basic information about learning resources, such as titles, descriptions, and categories. They often lack the capability to capture and represent the complex relationships between resources, teachers, students, and their peers. This limitation restricts traditional LMSs from fully *personalizing recommendations* based on interactions, such as a student's engagement with particular content, a teacher's preferred teaching methods, or collaborative activities among peers [55].

Alongside the common techniques mentioned above (namely CF, CBF and Hybrid) are applied, other research on RecSys use various alternative approaches based on the complex needs of educational domains. Ontology-based techniques in Knowledge-based RecSys can be used to enhance recommendation accuracy by structuring domain knowledge and identifying relationships between educational resources [30]. Additionally, educational question-routing in online student communities helps recommend relevant questions to students, encouraging them to answer and engage on MOOC platforms. This approach promotes knowledge sharing and builds a collaborative learning environment by connecting students with questions that align with their expertise and interests [26].

2.3 Natural Language Processing in Recommender Systems

NLP is a branch of artificial intelligence that enables computers to understand, interpret, and generate human language [44]. Recommender systems have traditionally relied on structured data, such as user ratings and item attributes. However, with the rising availability of textual data, NLP techniques are instrumental in processing and analyzing unstructured textual data (i.e. user preference statement). By extracting meaningful features from text, NLP calculates the similarities between users and items, enhancing the personalization and relevance of recommendations. This section reviews the role of NLP in recommender systems, focusing on feature extraction and similarity computation methods relevant to this thesis.

In Content-based Filtering, NLP is used to process textual item descriptions or user-generated content (e.g. preferences) to extract meaningful features that represent items. In other words, feature extraction is fundamental in NLP-based CBF, as it transforms raw text into structured representations that algorithms can process and compare. Word embeddings technique allows content-based filtering systems to understand the semantic similarity between items even when they do not share exact words [27]. For instance, a CBF system can recognize the similarity between "data science" and "machine learning" because their word vectors are close in the embedding space. Some applications of NLP-based CBF include:

- [29] discusses how NLP techniques are used to improve the accuracy and relevance of content-based book recommendations
- [3] discusses how word embeddings trained on Wikipedia contents can be utilized in CBF RecSys. The paper highlights how the NLP technique provide meaningful, semantical recommendations by capturing nuanced meanings of words.

CF is one of the most widely used approaches in recommender systems. However, CF methods often suffer from the data sparsity problem, where the user-item interaction matrix is sparse because users rate or interact with only a small fraction of available items. This sparsity results in less reliable recommendations of CF algorithms [1, 7].

The paper [19] proposes a recommendation model, Convolutional Matrix Factorization (ConvMF), which integrates a Convolutional Neural Network (CNN) into the matrix factorization technique. By doing so, ConvMF effectively captures the contextual information embedded in documents, leading to enhanced rating prediction accuracy. Extensive evaluations on three real-world datasets demonstrate that ConvMF significantly outperforms state-of-the-art recommendation models, even in scenarios where the rating data is highly sparse.

In this thesis, we employ NLP techniques to address the challenges posed by sparse data, due to the computational complexity of the CNN deep learning model.

2.4 Synthesizing training data

Synthetic data has gained attention as a viable solution to address challenges related to data availability, privacy, and diversity across various fields, including machine learning, healthcare, finance, and most importantly, recommender systems [23]. Unlike real data, synthetic data is artificially generated and designed to mimic the properties and distributions of real datasets without directly exposing sensitive information [34]. Essentially, one of the primary drivers behind synthetic data generation is data scarcity, particularly in domains where acquiring large amount of high-quality data is difficult such as in an educational domain. Furthermore, synthetic data usage in CF also resolves the problem of cold-start of new users or items by replacing user-item interaction data.

The paper [1] explains unique challenges in obtaining real-world cybersecurity data, particularly for training and testing cybersecurity models without risking sensitive information. It introduces rule-based technique, which is used to generate data according to a set of predefined rules to simulate realistic behaviors.

This research [46] discusses the use of Generative Pre-trained Transformer (GPT) models for generating synthetic data in the medical domain, where data scarcity and privacy concerns are major challenges. The use of GPT model, particularly ChatGPT API's synthetic data generation helps the development of more robust, flexible and complex language patterns within medical terminology and context.

In the educational setting of this thesis, we combine rule-based approach with GPT-based techniques to create synthetic data that is both contextually relevant and flexible. The rule-based approach ensures adherence to specific educational structures, guidelines, and patterns, while the GPT-based generation adds diversity and realism. This hybrid approach allows us to generate data that follows educational patterns expected from a user survey.

Chapter 3

Recommender Systems

"On average, the adult human makes 35,000 decisions a day."
— Bryan Robinson, Ph.D., *Forbes* [42]

Making decisions are activities that span across different aspects in our daily lives, from entertainment activities, e.g., deciding what to eat, what to watch on the television, to professional matters, e.g., which companies to apply, who among the applicants to accept, and specifically related to this thesis, where to pursue a Master's Degree. With the massive growth of content available, it is an increasing struggle for people to make decisions – a phenomenon known as the *paradox of choice* [45]. Recommender Systems are applications that attempt to mitigate this problem.

3.1 Definition

RecSys, in their simplest form, are applications that attempt to predict and "recommend" items in which applications' users may be interested. Today, recommender systems are embedded into our lives, often without people realizing the impact they have. This dependence on recommendations across vast domains highlights again the *paradox of choice.*

Example 1. Streaming services, like Netflix and Spotify, use RecSys to suggest movies, series, or songs based on users' viewing and listening history.

Example 2. E-commerce companies, like Amazon, leverage RecSys to suggest commercial products suited to users' past behavior and browsing interactions.

Example 3. Social media platforms, such as Facebook and Instagram, use RecSys to narrate news feed aligned with users' interests.

Example 4. Shopping applications, such as UberEats, and travel and hospitality services, such as Booking.com and Airbnb, also implement RecSys in their ecosystem.

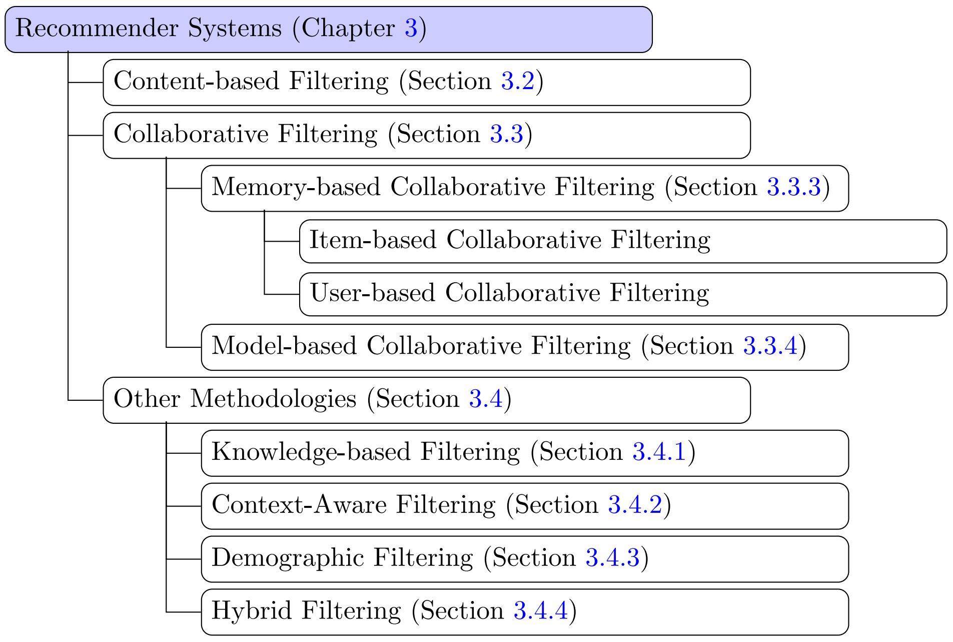

Recommender Systems (Chapter 3)
Content-based Filtering (Section 3.2)
Collaborative Filtering (Section 3.3)
Memory-based Collaborative Filtering (Section 3.3.3)
Item-based Collaborative Filtering
User-based Collaborative Filtering
Model-based Collaborative Filtering (Section 3.3.4)
Other Methodologies (Section 3.4)
Knowledge-based Filtering (Section 3.4.1)
Context-Aware Filtering (Section 3.4.2)
Demographic Filtering (Section 3.4.3)
Hybrid Filtering (Section 3.4.4)

Figure 3.1: Overview of popular recommendation methodologies

In an educational setting, prospective students, looking for (higher) education programs, such as a Master's Degree or a Doctorate Program, face an abundance of options worldwide, each with its own unique curriculum, subjects, study approaches and requirements. Selecting the right program without feeling like you are missing opportunities elsewhere can be a stressful and time-consuming experience.

RecSys offers a solution by helping students efficiently sort through thousands of options to find programs that match their academic backgrounds, career goals, and personal preferences. This thesis focuses on developing and evaluating RecSys to support students in making one of their most significant decisions of their lives: choosing a master's degree program.

RecSys employs different advanced recommendation algorithms (also known as filtering techniques). Each type of recommendation filtering techniques has its unique characteristics, strengths, and limitations, which consequently influence how it interprets the data. Figure 3.1 gives an overview of the most common filtering techniques.

In what follows, we discuss further these techniques. Our primary focus will be on Content-based Filtering (CBF, Section 3.2), and Collaborative Filtering (CF, Section 3.3). Within CF, we will delve into specific methods including Item-based Collaborative Filtering, Nearest-neighbor algorithms, Matrix factorization, and Singular Value Decomposition (SVD). While we recognize the importance of Knowledge-based Filtering, Hybrid RecSys, context-aware and demographic recommendations in the broader landscape, our discussion of these methods will be limited, i.e., we will only attempt to provide a brief overview to contextualize our main areas of study (Section 3.4).

3.2 Content-based Filtering

3.2.1 Definition

Content-based Filtering is a recommendation technique that, as the name suggests, uses contents – or in machine learning term, features – of an item to identify what a user may find appealing. It analyzes the characteristics of items, often in terms of keywords, text, or metadata, and then matches them with a user's preferences. The **key principle** of CBF, as shown in Figure 3.2, is that, if a particular item has been proven to be appealing to a user, similar items based on the content features of the item will likely be of interest to that user.

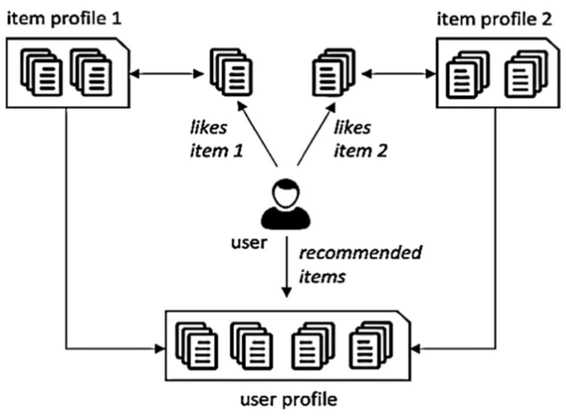


Figure 3.2: Content-based Filtering in Recommender Systems (taken from [43])

Example 5. Streaming Services recommend its items, such as videos, movies, or shows that are similar to those the user has watched previously. For instance, if a user watches a lot of crime thrillers, Netflix may suggest other crime-related content.

Example 6. News Platforms recommend articles based on topics the user has read before. If a user frequently reads articles on technology, the platform will suggest similar articles about tech advancements.

Example 7. Social media platforms, such as Facebook and Instagram, suggest posts based on what topics that you may have liked before.

In the context of Education, this approach has been applied for the course recommendation system. For example, on the Massive Open Online Course (MOOC) platforms like Coursera or Khan Academy, if a student has expressed an interest (or has enrolled) in Data Science programs, CBF would recommend *likewise* programs, potentially using keywords or attributes such as ”Machine Learning”, ”Artificial Intelligence” or ”Data Analytics” of the program features.

3.2.2 Discussion

CBF is a powerful method. In what follows, we discuss its advantages and limitations.

Advantages of Content-Based Filtering

- **No dependency on other users**: CBF only needs data on the user’s own interactions or preferences, making it effective even if there’s no interaction data from other users. Furthermore, this makes the system that uses CBF to scale easier to a larger number of users.
- **Highly personalized recommendations**: Since recommendations are generated based on individual’s preference, CBF produce very personalized suggestions that align closely with user.
- **Variability**: The model can capture the specific interests of a user, and can recommend “hidden gems” that only few users know.

Drawbacks of Content-Based Filtering

- **Over-specialization**: CBF may result in recommendations that are too similar to items the user has already engaged with, leading to limited variety and fewer opportunities for the user to discover new types of items. This is sometimes referred to as the “filter bubble” effect.
- **Limited discovery of new items**: Similarly, since the system recommends items similar to those the user has already shown interest in, it may not recommend items that fall outside of the user’s current preferences; therefore, the RecSys has limited ability to expand on the users’ existing interests and preferences.
- **Feature dependency**: CBF relies heavily on item features. If features are poorly defined, irrelevant, or lack sufficient detail (as may be the case with sparse descriptions), the quality of recommendations may be downgraded.

Overall, CBF is useful for domains where item characteristics are well-defined, and users have specific, consistent preferences. However, it has limitations in terms of variety and discovery, as it tends to recommend items similar to what the user already likes. In the context of recommending master’s programs, CBF could analyze program descriptions and match them with user-stated interests or background, making it an effective tool to personalize suggestions in educational settings.

In this thesis, the methodology discussed for CBF is limited to the NLP-based technique, which will be discussed further in Section 4.3.

3.3 Collaborative Filtering

3.3.1 Definition

The term "Collaborative Filtering" was first coined in 1992 by the developers of the Tapestry system, an early CF email system create at Xerox Palo Alto Research Center [12]. Tapestry was used to rate and annotate messages, enabling the system to filter and recommend messages to other users based on collective input. The core idea is based on the observation that *users who have agreed in the past usually agree again in the present and the future*, allowing the system to make personalized recommendations by analyzing patterns across user-item interactions.

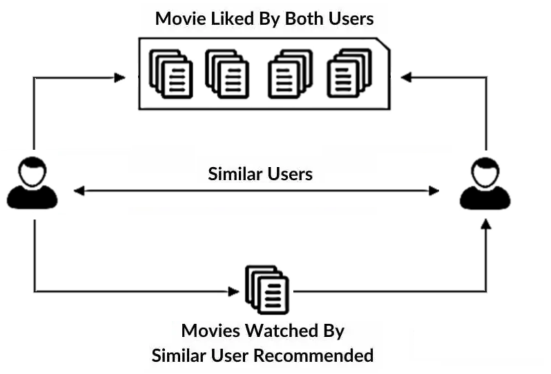

Movie Liked By Both Users
Similar Users
Movies Watched By Similar User Recommended

Figure 3.3: Collaborative Filtering in Movie Recommender System

Example 8. Figure 3.3 displays an example of how CF works in the context of movie recommender system. For instance, Netflix streaming service offers a feature *"Because You Watched"* in order to recommend similar film/movies based on what the users have previously watched.

Example 9. E-commerce platforms – such as Amazon, have a so-called *"customers who bought this item also bought"* feature to analyze items that are frequently bought altogether. In addition to that, if a user purchases or view a product, they have a *"Recommended for you"* section that will suggest products common to the one user has viewed. This enhances the user experience, hence improve sales output.

Example 10. Personalized Spotify's analytics feature *"Discover Weekly"* displays the listening history and compare it with users who have similar tastes in music.

3.3.2 Discussion

Before delving into specific approaches, it is essential to understand the strengths and limitations of collaborative filtering as a whole.

Advantages of Collaborative Filtering

- **Domain independence**: CF does not require an understanding of item content, making it versatile across different domains without the need of having specialized knowledge or complex content processing.
- **Discovery of new interests**: This tackles one of the biggest issue of limited expansion of users' existing preferences, CF introduces users to items they might not find through content-based filtering method, increasing user engagement through unexpected but relevant suggestions.

Drawbacks of Collaborative Filtering

- **Cold-start problem**: This occurs when users or items are newly updated to the system, leading to a sparsity of information [24]. For new users, without prior interaction data, the system cannot accurately recommend items to new users. In CF, this remains the biggest problem as new product or item at the time of introduction has not been rated or interacted with; as a consequence, the system lacks data to associate them with any user.
- **Data sparsity**: In practical applications, the user-item interaction matrix or matrices tend to be rather sparse, since users usually provide ratings for only a small percentage of the items [17]. This leads to a difficulty in finding significant correlations between users and items.

In general, CF disregards the specific features of an individual item - and instead, concentrates on user interactions with items in the form of ratings. By analyzing patterns in these interactions across many users, we identify relationships and factors that help predict preferences and make recommendations. There are two main types of collaborative filtering approaches: memory-based (Section 3.3.3) and model-based (Section 3.3.4).

3.3.3 Memory-based Collaborative Filtering

Unlike content-based methods, which utilize the features of items, this approach only relies on the historical interaction data between users and items to make recommendations, usually denoted as the User-item Interaction Matrix (Example 3.1).

User-item Interaction Matrix is, a two-dimension matrix (one dimension is for the item, the other is for the user), in which each cell represents a user's rating of an item. Commonly, the type of the rating (i.e., the type of the cell) is positive number with an additional symbol "?" indicates a missing value for an item with which a user has not yet interacted. Rather than building complex models, Memory-based CF focuses on finding similarities between users or items based on their previous known interactions. It can be further categorized into two main techniques: user-based and item-based.

User-based Collaborative Filtering

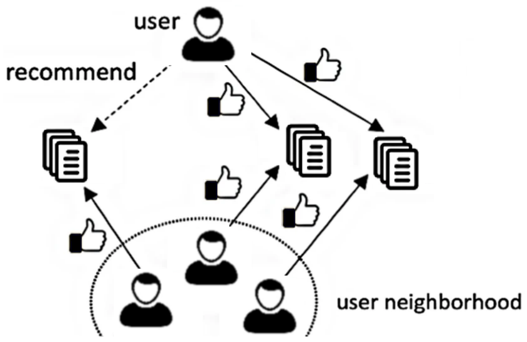


Figure 3.4: User-based Collaborative Filtering Example (taken from [43])

This technique finds users with similar interaction patterns to a target user and recommends items that these similar users have engaged with. It assumes that users with shared preferences in the past will continue to have similar tastes, allowing recommendations based on the interactions of similar users.

	Item 1	**Item 2**	**Item 3**	**Item 4**	**Item 5**
`User A`	5	3	4	4	?
`User B`	3	1	2	3	3
`User C`	4	3	4	3	5
`User D`	1	3	1	?	2

Table 3.1: Example of a User-item Interaction Matrix (1: lowest to 5: highest)

Example 11. For example, in Table 3.1, in order to determine whether **Item 5** would be a good recommendation for `User A`, we first look for users who share similar preferences with `User A`. In this case, let us say that `User A` have similar tastes with `User C` because both have rated Item 2 and 3 similarly. Additionally, they both enjoy **Item 1** and **Item 4**, with moderately high ratings. Given that `User C` rates **Item 5** highly and has similar ratings to `User A` on multiple items, it is statistically reasonable to say `User A` would likely enjoy **Item 5**.

Item-based Collaborative Filtering

In contrast to User-based CF, Item-based CF identifies *items* that are similar based on user interactions. It assumes that users are keen on enjoying items similar to those they have already liked.

Example 12. In Table 3.1, let us examine how we can predict how `User D` would rate **Item 4**. In this case, we first identify that **Item 1** and **Item 3** share close ratings with **Item 4** across multiple users. `User A` rated **Item 1**, **Item 3**, and **Item 4** as 5, 4 and 4, respectively, while `User C` rated them 4, 4 and 3. This proximity suggests that **Item**

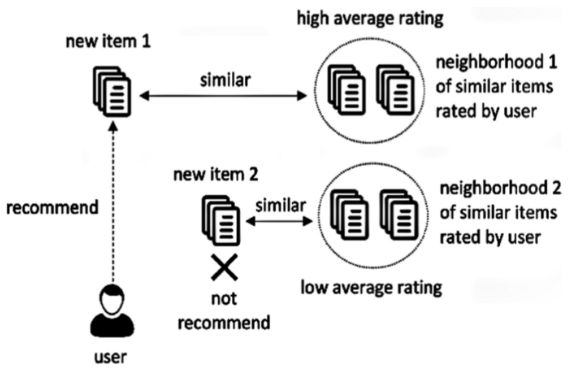


Figure 3.5: Item-based Collaborative Filtering Example (taken from [43])

1 and **Item 3** may be good indicators of how `User D` might rate **Item 4**. Next, we observe `User D`'s ratings for **Item 1** and **Item 3**. `User D` rated **Item 1** as 1 and **Item 3** also as 1, indicating a generally low interest in items similar to **Item 4**. Given this pattern, we can infer that `User D` would likely rate **Item 4** similarly, with a low score.

Discussion

Memory-based CF in general offers straightforward approach, which makes it easy to understand and implement. One of the key advantages of memory-based collaborative filtering is its *transparency*, recommendations are explainable based on observable patterns in the data, so it is easy to justify why a particular item was recommended.

However, performance degrades with large datasets due to increased computational demands. For example, consider a scenario where a recommender system manages a dataset with one million items. When a new item is added to this dataset, the system must evaluate its similarity to each of the existing items to provide accurate recommendations. This requires computing one million additional comparisons. This scalability issue can lead to delayed recommendations and require substantial computational resources, which may not be practical in real-time applications.

To address poor scalability, model-based methods are preferred in larger-scale systems, as they reduce the complexity by learning latent factors and optimizing data processing.

3.3.4 Model-based Collaborative Filtering

Model-based CF employs advanced mathematical and machine learning models to predict user preferences. Instead of using the raw interaction data directly, it builds a predictive model that can generalize from the observed data to make recommendations. Popular techniques include *Non-Negative Matrix Factorization* and *Singular Value Decomposition*. These approaches work by breaking down large datasets into simpler representations, capturing hidden factors that reflect user interests and item characteristics. For instance, NMF reduces a complex user-item interaction matrix into lower-dimensional matrices, helping the system predict which items a user might like based on shared characteristics with other users or items. By leveraging these latent factors, NMF can handle large datasets more efficiently while deliver accurate recommendations even with sparse data.

As previously stated, Model-based CF offers several advantages, particularly in its ability to handle large datasets and uncover complex, hidden patterns in user preferences. One primary benefit is its ability to handle large datasets with greater scalability and efficiency compared to memory-based methods. By reducing the original user-item interaction matrix into lower-dimensional representations, model-based approaches can perform well even when the dataset is sparse. For instance, matrix factorization techniques significantly improved the performance and scalability of recommender systems used in platforms like Netflix [20].

Despite their strengths, one major drawback is the increased computational complexity associated with training the models. Matrix factorization require iterative optimization algorithms, which can be computationally intensive and time-consuming, especially with very large datasets. This complexity demands substantial computational resources and can make real-time updating of the model challenging when new data arrives frequently [20].

In summary, both memory-based and model-based approaches have their unique advantages and challenges. Memory-based methods are simple and transparent but struggle with scalability and data sparsity. Model-based methods, on the other hand, provide improved accuracy and handle large datasets effectively.

3.4 Other Relevant Methodologies

3.4.1 Knowledge-based Filtering

Knowledge-based recommender systems offer an alternative approach by utilizing domain knowledge about users and items to generate recommendations. Unlike CF and CBF, knowledge-based systems do not rely on historical user-item interactions. Instead, they focus on understanding the needs of the user and how specific item features can satisfy those needs.

Example 13. In a real estate search platform, when a user is looking for a property, they input explicit preferences such as location, budget, number of bedrooms, property type, proximity to schools or public transport. The system uses this information to filter and recommend properties that match the user's criteria.

Since Knowledge-based systems do not depend on prior user interaction with the data, they do not suffer from the cold start problem. The systems can provide accurate recommendations to new users or items without the need for initial ratings. In addition, recommendations are made based on explicit rules and knowledge, therefore, users can understand why certain items are recommended.

On the other hand, the process of obtaining, building and maintaining the knowledge requires significant effort and substantial domain understanding. Furthermore, the users should not expect "hidden gem" of items since the systems are built to look something specific rather than something unexpected but pleasant.

In the context of this thesis, which focuses on developing a recommender system for students selecting academic programs, knowledge-based recommender system would be a powerful methodology to consider. Students often have specific criteria when choosing a program, such as:

- Field of Study: Specific majors or interdisciplinary programs.
- Location Preferences: Geographic areas, campus settings, or online options.
- Admission Requirements: Admission prerequisites, standardized test/examination scores, or language proficiency.
- Career Goals: For example, programs with strong industry connections or with various internship opportunities.

For example, if a student is interested in pursuing a Master's degree in data science with a focus on machine learning, prefers a university in Europe, and wants an internship opportunity, the system can apply this information to recommend suitable programs that meet all these criteria. This approach ensures that recommendations are not only relevant but also aligned with the specific goals and constraints of each student, ultimately contributing to more informed and satisfactory educational decisions.

3.4.2 Context-Aware Filtering

Basically, context-aware recommender systems incorporate contextual information into the recommendation process, offering unique value to other RecSys. While traditional approaches in recommender systems rely solely on the user preferences expressed as item ratings, those recommendations do not take into consideration that user change their preferences over a period of time [22]. Not only that, the context can include factors like location, weather, mood, device type, or social environment. By considering context, these systems aim to provide more relevant and timely recommendations that align with the user's current situation.

Example 14. If a student is finding materials to study while traveling from home to school using only a smartphone, context-aware recommendations should prioritize audiovisual content rather than lengthy, text-dense documents.

Example 15. Location-based recommendations suggest nearby places or attractions based on the user's current location and time of day. Also, the system should suggest indoor and outdoor activities based on the weather condition.

Nevertheless, the process of collecting and processing contextual information may raise privacy issues. For instance, Google Maps has faced concerns regarding user location data privacy [6], Fitness app Strava accidentally discloses the location and shape of secret US army bases [49], or plane tracking application discloses individual's private travel plans [50].

In an educational setting, incorporating context-aware features would enable the system to make recommendations that reflect students' current needs, behaviors, and program availability. For example, the system could provide tailored recommendations based on factors like application deadlines, geographic location, and program availability, ensuring students receive suggestions that are both personalized and - at the time, relevant.

3.4.3 Demographic Filtering

Demographic filtering uses demographic information about users — such as age, gender, income, education level, or occupation — to make recommendations. The underlying assumption is that users with similar demographic profiles have similar preferences. This method segments users into groups and recommends items popular within those groups. Unlike collaborative filtering, demographic recommenders do not rely on users' data, making them useful when user - item interaction is scarce or unavailable. In addition, they are particularly helpful for new users (cold-start users) where no prior behavior data exists.

Example 16. Targeted advertising (i.e. online advertisements) are displayed based on age group, gender and other demographic aspects to increase relevance to customers.

Example 17. Retail stores suggest products that are popular among a certain demographic group.

In relation to the research, demographic filtering can provide useful insights into user preferences based on general characteristics such as educational background, country of residence, or language preference. A student from a particular country may favor programs closer to home or those with faculty who speak their native language. Similarly, educational background can influence preferences, with students from technical fields possibly favoring programs with a research or technical focus.

However, by grouping users into demographic segments, recommendations may be overly generalized and fail to account for individual preferences within each group. This also leads to stereotypical assumptions regarding gender or age group, which may even lead to more biased or inappropriate recommendations. Collecting and using demographic data can raise privacy concerns. Demographic recommender systems have not gained widespread popularity, largely due to security and privacy issues that contribute to users' hesitation. Consequently, obtaining accurate demographic data from users poses a significant challenge.

3.4.4 Hybrid Recommender Systems

Generally speaking, hybrid recommender systems are not a novel approach; rather, they represent an optimized combination of existing recommendation methodologies. By integrating methods such as CF, CBF. and other filtering techniques, Hybrid Recommender Systems aim to leverage their individual strengths and mitigate their weaknesses [17], resulting in more superior and accurate performance than any single component techniques [5]. It is also possible to combine different techniques of the same type of RecSys [54].

Example 18. Streaming Services: Platforms like Netflix employ hybrid systems to suggest media based on user history, demographic data, and content similarity. Netflix's hybrid recommendation system effectively addresses the limitations of individual approaches. This allows them to provide more personalized and diverse recommendations, increasing overall user experience.

Combining multiple methods, such as collaborative filtering, content-based filtering, and demographic filtering, could provide a more advanced recommendation experience for students seeking graduate programs. For instance, CF can suggest programs that similar users have shown interest in, while content-based filtering ensures that program attributes align with a student's academic goals. Demographic filtering can also play a role by providing relevant initial recommendations based on background factors such as location or educational history, particularly useful for new users with minimal interaction history. Hybrid systems capitalize on the strengths of each technique to overcome individual limitations and deliver more well-rounded recommendations.

Despite that, hybrid systems can be difficult to interpret, as it is not always clear how recommendations are derived when multiple algorithms are involved. Additionally, hybrid systems can be computationally intensive and complex to implement, requiring significant resources for algorithm integration, tuning, and testing.

Summary This section has provided a comprehensive overview of recommender systems, delving into their fundamental principles and various methodologies. We explored how collaborative filtering leverages user-item interaction data to generate personalized recommendations, focusing on both memory-based and model-based approaches.

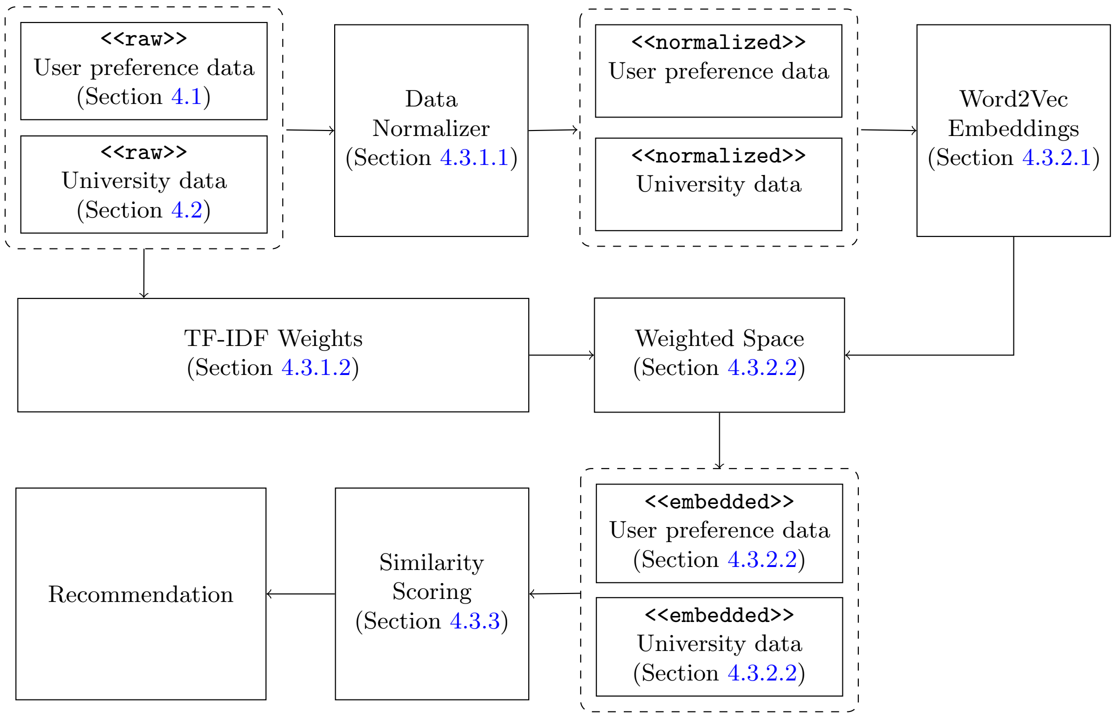
Content-based filtering was discussed as an alternative that relies on the attributes of items and user preferences to suggest similar items. Additionally, we briefly examined other techniques such as knowledge-based, hybrid, context-aware, and demographic filtering, each with its unique advantages, limitations and suitable application contexts.

While numerous techniques exist, this thesis will concentrate on *developing an NLP-based Content-based Filtering* as well as *analysis of different techniques within Collaborative Filtering*. This comparison will provide valuable insights into how these key techniques can be optimized and effectively utilized in the design of recommender systems.

Chapter 4

Methodology

To begin with, Figure 4.1 describes an overview of our methodology.



<<raw>> User preference data (Section 4.1)
<<raw>> University data (Section 4.2)
Data Normalizer (Section 4.3.1.1)
<<normalized>> User preference data
<<normalized>> University data
Word2Vec Embeddings (Section 4.3.2.1)
TF-IDF Weights (Section 4.3.1.2)
Weighted Space (Section 4.3.2.2)
Recommendation
Similarity Scoring (Section 4.3.3)
<<embedded>> User preference data (Section 4.3.2.2)
<<embedded>> University data (Section 4.3.2.2)

Figure 4.1: Data collection and recommendation methodology

In a nutshell, we first synthesize the user preference data (Section 4.1) using a machine learning model. Then, we using a web-scrapping technique to collect study programs of different university (Section 4.2). Together these data form the input `raw` data of our methodology. This `raw` data is used as an input for several applications those are described in Section 4.3.

- Firstly, it is used to generate the Weighted Space (Section 4.3.2.2) using a word embeddings technique called `Word2Vec` [9] (Section 4.3.2.1). It is worth mentioning that before this step, we also perform some normalization on the `raw` data (Section 4.3.1.1), which results in `normalized` data.

- Secondly, it is used to construct the vocabulary weights for the TF-IDF model (Section 4.3.1.2).

The latter output serves as an input of the former output, this results in `embedded` user preference and university data. With the `embedded` data, we perform a similarity scoring calculation (Section 4.3.3) to complete our methodology.

4.1 Synthesizing User Preference Data

4.1.1 User Preference: Definition

In our approach, as a proof of concept, we synthesize user preferences in the form of English sentence(s) representing a single user's preference.

No.	**User preferences**
1	I want to study Data Science in Germany for two years.
2	Looking for an online Master's in Business Administration with a focus on entrepreneurship.
3	Interested in a Master's program in Renewable Energy Engineering taught in English.

Table 4.1: Synthetic User Preference Examples (🟨 highlights the specific preferences)

Table 4.1 depicts a few examples of synthetic user preferences with different complexity.

1. In the first query, the user (i.e., student) is looking for a study program for Data Science (i.e., study fields), in Germany (i.e., personal preferences – place of study), for two years (i.e., personal preferences – program duration).

2. In the second query, the user is looking for an online (i.e., personal references – mode of study) Master's (i.e., program level) in Business Administration (i.e., study fields) with a focus on entrepreneurship (i.e., personal preferences – program focus).

3. In the third query, in addition to be a Master's program (i.e., program level) in Renewable Energy Engineering (i.e., study fields), the user queries for the program that is taught in English (i.e., language of instructions).

4.1.2 Advantages of Synthetic User Preferences

Synthetic user preferences can replicate real user input.[1] This approach has several advantages. On one hand, this approach is *time-effective*: there is no need to spend

[1]In Chapter 5, we provide some empirical evidence to back this claim up

time collecting data (which would result in creating surveys, collecting and cleaning the answers), and *ethical-compliance*: since synthetic data are not personal, this approach ensures full compliance with data protection regulations.

On the other hand, this approach offers diverse generation of user preferences, spanning across various study fields, program durations, and other specific requirements. Since universities typically offer a finite number of programs, this approach could potentially be applied for different domains beyond the scope of the university's programs. As a result, this benefits students, the primary users of RecSys for Education, by expanding their search options.

4.1.3 Limitations of Synthetic User Preferences

One limitation of using synthetic user preferences is the lack of realistic behavioral data as these simulated data cannot fully capture the complexity of real user preferences. Particularly, for recommender systems filtering techniques that rely on the users interaction with the system (e.g., user-based CF), this approach would potentially downgrade the performance of those systems.

4.1.4 Data Synthesis Methodology

In this thesis, we synthesize user preferences using prompt engineering techniques. More specifically, we rely on the ChatGPT of OpenAI [32], the state-of-the-art commercial large language model (LLM), and its application programming interface (API) to synthesize list of user preference sentences for querying the Master's programs. The following snippet (in Figure 4.2) displays our final, refined prompt message:

4.1.5 Output

Appendix A displays the prompt output examples for user preferences for Master's programs in which $N = 50$.

In our work, we use these output generated data as an input to the case studies to conduct evaluation on the realism of the synthetic data (Chapter 5) and on the effectiveness of the state-of-the-art recommender algorithms (Chapter 6). For the former study, we use the synthetic user preference as input of a academic survey. For the latter study, we use them as input data for the recommender systems being developed and tested.

While the survey of the former study is explained in detail in Section 5.2, we proceed to explain the methodology of the latter study in the next section.

```
Imagine you're a student exploring master's programs in Germany. Write a
  variety of natural-sounding statements that reflect your interests,
  aspirations, and potential constraints. Each statement must include
  either a specific program name (e.g., ‘Computer Science'), a general
  field of interest (e.g., ‘engineering'), or a degree type (e.g.,
  ‘Master of Science'), and ideally both. However, allow for ambiguity in
  any of these details to reflect openness to various options.

User Preferences:
  - Program Name: Include both specific programs (e.g., Computer Science)
  and general fields of interest (e.g., Engineering, Business). Allow
  for ambiguity by sometimes mentioning only the general area without
  specifying the exact program.
  - Location: Highlight a mix of prominent cities like Berlin, Hamburg,
  and Munich, as well as mid-sized university hubs like Freiburg or
  Gottingen. Encourage expressions of openness to multiple locations.
  - Degree Type: Include common options such as Master of Science, Master
  of Arts, and Master of Education, along with unique formats like
  modular programs, international courses, or accelerated degrees.
  - Language: Include programs taught in English and German, with
  occasional mentions of dual-language offerings or those tailored for
  non-native speakers.
  - Subject: Range from foundational areas (e.g., Engineering and
  Management) to interdisciplinary or emerging fields (e.g., Digital
  Transformation, Business Mathematics).
  - Study Mode: Include a balance of full-time, part-time, distance
  learning, and international study formats.

Task:
Generate N statements that align with these guidelines. Ensure diversity,
  clarity, and relatability in the statements, focusing on popular and
  emerging options. Introduce ambiguity by expressing openness or
  flexibility in any preferences, including program names, by sometimes
  using general fields of interest or unspecified areas. Avoid overly
  niche or obscure topics while maintaining a realistic and personalized
  tone.
```

Figure 4.2: Prompt message for synthesizing user preferences for Master's programs

4.2 Collecting Universities' Programs

The universities' programs dataset comprises information about academic programs offered by institutions. In this study, we focus on universities/institutions that are in Germany. This data is intended to serve as a foundation upon which recommendations are generated, enabling the generation of relevant academic program suggestions based on user preferences. This subsection provides a detailed description of the dataset and outlines the methodology employed in its collection.

4.2.1 Data Descriptions

The data was extracted from [15] - a university portal in Germany, which provides detailed information on academic offerings across various institutions. This portal was selected for its extensive and detailed coverage of university programs in the country, making it a helpful resource for this research. The attributes collected from the data portal are as described in Table 4.2. By leveraging these attributes, the recommendation system can effectively match user profiles and preferences with suitable academic programs.

4.2.2 Web Scraping Techniques

The methodology employed to gather data on universities' programs involved the use of web scraping techniques. Web scraping is a process by which information is extracted from websites and transformed into a structured format suitable for analysis [25]. In this case, data from [15] was parsed through the web scraping technique to collect program features and store in a Comma-separated values (CSV) format. This approach was used to compile data more efficiently from publicly accessible online resources.

To extract the necessary information from the university portal, HTML Parsing techniques were used. HTML parsing is a critical step in web scraping, where the raw HTML content of a webpage is systematically parsed to identify and retrieve specific elements that contain relevant data. The extraction process involves navigating the HTML structure of the web pages and selecting the specific tags (e.g., `<h1>`, `<div>`, `<span>`) that correspond to the data of interest, such as program names, descriptions, and admission requirements.

In this study, tools such as BeautifulSoup [41] and Selenium [36] were used in the scraping process. Selenium WebDriver was useful for cross-browser testing and automating browser interaction, as it allowed the automated downloading of dynamic HTML content. Once the HTML content was successfully retrieved using Selenium, BeautifulSoup was employed to parse the static HTML content. By targeting specific tags , the relevant information for each program description was extracted and stored in a structured CSV format, allowing for later use in CBF approach.

Attribute	**Description and Examples**
Program Name	Title of the academic program. *Examples:* Computer Science, Urban Planning.
University	Institution offering the program. *Examples:* Technical University of Berlin, University of Cologne.
Location	Geographical location of the university. *Examples:* Berlin, Cologne.
Duration	Length of the program. *Examples:* 4 semesters, 32 months.
Degree Type	Type of degree awarded upon completion. *Examples:* Master of Science, Master of Arts.
Language	Language of instruction. *Examples:* English, German.
Subject	Focus of the academic program. *Examples:* Statistics, Software Engineering.
Study Mode	Whether the program is online, on-campus, or hybrid. *Examples:* Online, On-campus, Hybrid.
Overview	A detailed overview of the university or program. *Example:* offers a strong focus on interdisciplinary studies; providing cutting-edge research opportunities; state-of-the-art facilities.
Teaching & Studying	Description of teaching methods and program structure. *Example:* program combines lectures, hands-on workshops, and collaborative projects; ensure a balance of theoretical and practical skills.
Researching	Information about current research projects and facilities. *Example:* research focuses on a certain field/discipline.

Table 4.2: University Portal Description

It is worth denoting that the process of parsing information from [15] was conducted in compliance with data copyright policies. Only publicly accessible data was collected, respecting the ethical guidelines of web scraping. Additionally, in accordance with the terms of use, this data was utilized exclusively for academic purposes and excluded from any commercial use. Since the use of the extracted information for research purposes only, the data collection process ensured both ethical integrity and compliance with legal standards.

4.3 NLP-based Content-based Filtering

As previously discussed in Section 3.2, CBF relies on the attributes of items themselves rather than user interactions or ratings. The focus of this section is on the practical application of CBF to university program recommendations, particularly when explicit user ratings are absent. By extracting detailed features about program names, descriptions, and other relevant metadata collected in Section 4.2, and combining with user preferences as outlined in Section 4.1, the system can generate recommendations based on the similarities between programs and user preferences.

The next subsections will discuss further how features are extracted from both dataset and how how similarity scores are calculated.

4.3.1 Feature Extraction and Similarity Scoring

4.3.1.1 Data Normalization

Before extracting features from the dataset, the raw universities' program descriptions and user preferences datasets went through extensive preprocessing steps. The datasets, as described earlier (Section 4.1 for User Preference, Section 4.2 for Universities' Program), consisted entirely unstructured (i.e., `<raw>`) textual data. Thus, data normalization process is required to ensure data consistency and uniformity for feature extraction.

Figure 4.3 denotes key steps in the data normalization process, which includes:

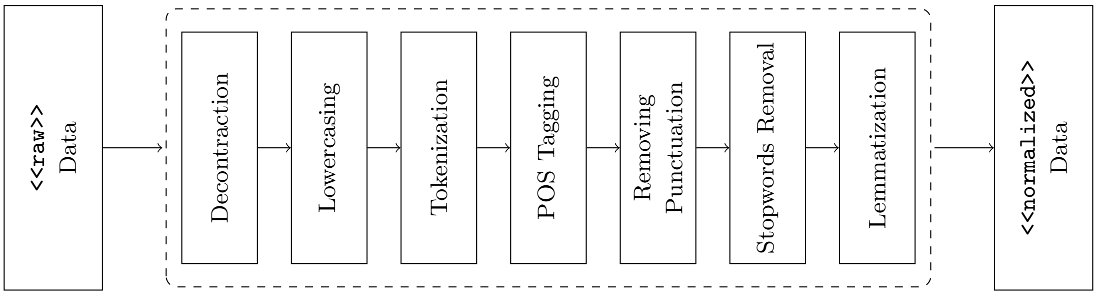


Figure 4.3: Data normalization process

- **Decontraction**: contractions within the text were expanded to their full forms to standardize the vocabulary. For example, “can’t” was converted to “cannot“
- **Lowercasing**: text was converted to lowercase to maintain uniformity and prevent the same word in different cases (e.g., “Program” and “program”) from being treated as separate entities.
- **Tokenization**: data was broken down into individual words called tokens, which allowed for detailed analysis of each term within the program descriptions. For instance, the sentence “Data Science is essential” would be tokenized into [“data”, “science”, “is”, “essential‘”.
- **Part-of-speech (POS) Tagging**: after tokenization, each token was assigned a grammatical category (i.e., Noun, Verb).
- **Removing Punctuation**: all punctuation marks (i.e., commas, periods, exclamation marks) were removed from the textual data, since they do not contribute to the semantic context.
- **Stopwords Removal**: similar to punctuation, stopwords (i.e., “the”, “a”, “is”) are commonly used words that contribute little to semantic value. Hence they were removed to focus on key terms that provide more insights into textual descriptions.
- **Lemmatization**: ensures that different grammatical forms of a word are treated as a single entity. For example, tokens, such as “studying” “studies” and “studied”, were all converted to their base form: “study“

Example 19. For example, consider the user preference query:

“I’m looking for a Master of Science in Computer Science, ideally in Berlin, where I can gain exposure to cutting-edge AI technologies.”

After applying the normalization process, the output becomes:

```
[‘interested’, ‘master’, ‘science’, ‘computer’, ‘science’,
‘berlin’,‘gain’, ‘exposure’, ‘cutting-edge’, ‘ai’,
‘technology‘]
```

This transformation removes extraneous elements, such as punctuation and common stopwords (e.g., “I’m”, “in”, “to”), and retains only the most meaningful words. Additionally, terms were lemmatized to their base forms (e.g., “technologies” to “technology”). These preprocessing steps ensured that the text data was standardized, denoised, and transformed into a structured format, ready for further processing.

4.3.1.2 TF-IDF Weighting

Term Frequency-Inverse Document Frequency is a statistical method commonly used in text analysis to evaluate the importance of a term to a document in a collection of documents or corpus [21, 37]. TF-IDF is composed of two components: Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency measures the frequency of a term in a document, emphasizing its prominence within that text. Inverse Document Frequency, on the other hand, reduces the weight of terms that appear frequently across the corpus, as such terms are less likely to provide meaningful differentiation between documents. The TF-IDF weight of a term t in a document d is mathematically expressed as:

$$\mathrm{TF\_IDF}(t,d) = \mathrm{TF}(t,d) \cdot \mathrm{IDF}(t)$$

where

- $\mathrm{TF}(t,d)$ is the frequency of term t in document d, and
- $\mathrm{IDF}(t) = \log\left(\frac{N}{n_t}\right)$, where N is the total number of documents in the corpus, and n_t is the number of documents *containing the term* t.

In the context of this work, TF-IDF plays a pivotal role in extracting meaningful features from both universities' dataset and user preferences to enable CBF approach for generating university program recommendations. TF-IDF is applied to the program descriptions and user profiles to assign weights to terms based on their relevance.

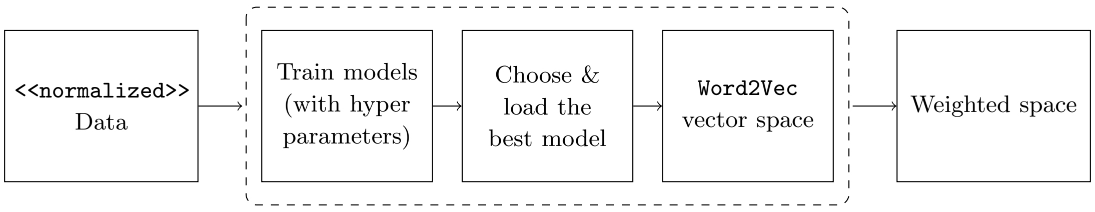
4.3.2 Feature Representation

4.3.2.1 Word2Vec Word Embeddings

`Word2Vec`, as a neural embedding model, is a technique that generates vector representations of words based on their semantic context [9]. Basically, the idea of `Word2Vec` is to represent each word as a multi-dimensional vector, where the position of the vector in the vector space captures the meaning of the word. For example, terms like "computer" and "science" are mapped closely in the vector space due to their frequent co-occurrence in similar contexts.

While `Word2Vec` excels at capturing semantic similarity, it assigns equal importance to all terms. This can be problematic when using pre-trained models in domain-specific applications, as they might allocate similar vector representations to words that are unrelated to those in the academic domain. To address this, a custom, domain-specific `Word2Vec` vocabulary mitigates this issue by anchoring word representations at hand. Figure 4.4 demonstrates how a custom `Word2Vec` model is built.

In this research, the custom vocabulary is derived from the *normalized* datasets, as discussed in Section 4.3.1.1. By restricting the vocabulary to terms frequently encountered in this domain, the embeddings become more precise in capturing relationships between words such as “artificial” - “intelligence”, “machine” - “learning” and “data” -“science”. Figure 4.4 shows how the model is trained on a domain-specific corpus, generating word embeddings that capture the semantic relationships between terms relevant to the academic context. The output of this model serves as the foundation for generating weighted sentence embeddings talked about in Section 4.3.2.2.



<<normalized>> Data
Train models (with hyper parameters)
Choose & load the best model
Word2Vec vector space
Weighted space

Figure 4.4: Process of Calculating Vector Space from Normalized Data in `Word2Vec`

In the Python implementation, the `Word2Vec` model from the *Gensim* library was used to generate word embeddings from the dataset. Gensim provides an efficient and flexible implementation of `Word2Vec` [57], allowing for customization of key *hyperparameters* to optimize the embeddings for domain-specific tasks. The hyperparameter tuning process involved defining a parameter grid and exploring multiple configurations to identify the optimal settings for the embeddings, as denoted in Figure 4.5. Common semantically related word pairs, such as “machine” and “learning”, “data” and “science” or others were used to evaluate and fine-tune these hyperparameters, ensuring that the embeddings adequately represent domain-specific relationships.

```
model = Word2Vec(
   corpus,
   vector_size,          # dimensionality of the word vectors
   window,               # context window size
   min_count,            # ignores words with total frequency lower
   sg,                   # skip-gram (1) or CBOW (0) training method
   epochs,               # number of iterations
   alpha,                # initial learning rate
)
```

Figure 4.5: Parameters for Gensim `Word2Vec` Model

4.3.2.2 Weighted Sentence Embeddings

As mentioned earlier in Section 4.3.2.1, `Word2Vec` treats all words equally important, regardless of their relevance or appearance rate to a specific document or corpus. This raises an issue when constructing sentence embeddings, as terms with different importance within a document or domain are given the same weight in the final representation. A good example would be terms such as "university" which will appear infinitely bountiful in this, should have little to none value for vector representation of a sentence, may be overemphasized, while domain-specific terms such as "business administration", which carry more semantic weight, may not be adequately highlighted.

To address this limitation, TF-IDF weights (discussed in Section 4.3.1.2) are applied to the `Word2Vec` word embeddings to create weighted sentence embeddings. The weighted sentence embedding is computed as the weighted sum of the product of each word's embeddings and its corresponding TF-IDF weights:

$$\text{SentenceEmbedding} = \sum_{i=1}^{n} \text{TF\_IDF}(t_i) \cdot \text{Word2Vec}(t_i)$$

where

- t_i represent a term in the sentence
- $\text{TF\_IDF}(t_i)$ is the TF-IDF wight of term t_i
- $\text{Word2Vec}(t_i)$ is the `Word2Vec` vector of term t_i

This process results in sentence embeddings that are both semantically relevant, enabling the recommendation system to better align program descriptions with user preferences. These weighted sentence embeddings are subsequently used to calculate similarity scores between program descriptions and user profiles, as detailed below in Section 4.3.3.

4.3.3 Similarity Scoring

Once the feature representations for both university program descriptions and user preferences are constructed, the next step is to calculate similarity scores. These scores are crucial for ranking and recommending programs that are most aligned with a user's interests. In this research, similarity is measured by calculating the *cosine similarity* between the weighted sentence embeddings of program descriptions and user preferences.

Cosine similarity Cosine similarity calculates the cosine of the angle between two vectors in a high-dimensional space, with values ranging from 0 (completely dissimilar) to 1 (perfectly similar). The formula for cosine similarity is given by:

$$\text{CosineSimilarity}(A, B) = \frac{\sum_{i=1}^{n} A_i . B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where

- A represents the weighted sentence embedding of program description
- B represents the weighted sentence embedding of user preferences
- A_i, B_i are individual dimensions of each vector space. A higher cosine similarity score indicates a stronger match between a program and the user's preferences.

Application of Weighted Sentence Embeddings in Similarity Scoring The weighted sentence embeddings, constructed by combining `Word2Vec` word embeddings with their corresponding TF-IDF weights (as detailed in Section 4.3.2.2), serve as the input vectors for cosine similarity computation.

Cosine Similarity Computation Calculating cosine similarity for each user-item pair involves comparing the weighted sentence embeddings of a program description (as an item) with the embeddings of a user preference. This process generates a similarity score for each pair, quantifying the alignment between the program's features and the user's interests.

User-Item Ratings The output of this computation is a *user-item similarity matrix*, where each row represents a user, and each column corresponds to a program. This matrix not only serves as the basis for ranking the recommendations but also acts as a critical input for collaborative filtering techniques, which will be discussed further in Section 4.4.

Ranking and Recommendations The computed cosine similarity scores are used to rank university programs based on their relevance to the user's preferences. The ranking process involves:

- Sorting programs in descending order of similarity scores.
- Selecting the N top-ranked programs as recommendations.

4.4 Collaborative Filtering Techniques

CF, as introduced in Section 3.3, is a recommendation approach that relies only on an user-item interaction matrix. In this section, we delve into the application of CF techniques, emphasizing the use of synthetic data, the challenges of data sparsity, and the exploration of four primary CF methodologies.

4.4.1 Use of Synthetic User Rating

To evaluate and compare collaborative filtering algorithms effectively, in this research, we employed synthetic user rating data generated by mapping NLP-based CBF user-item ratings. We computed initial similarity scores between users' preferences and programs' attributes in Section 4.3.3. These similarity scores were then mapped to a standardized rating scale of 1 to 5 to create synthetic user ratings matrix. During this mapping process, we introduced a noise variable of $\pm X$ to randomize the synthetic ratings, acknowledging that user behavior is often unpredictable and influenced by multiple factors beyond the direct similarity between items and preferences. By introducing this noise, the synthetic ratings reflect the uncertainty and diversity found in real-world data.

4.4.1.1 Data Sparsity Testing

Data sparsity is a prevalent challenge in CF, arising when users rate only a small subset of available items. This sparsity of explicit user ratings can defer the performance of CF algorithms to identify meaningful patterns and similarities. To investigate the impact of data sparsity on algorithm performance, we conducted experiments by adjusting the proportion of known ratings within the synthetic dataset.

By varying the data density—from sparse scenarios (e.g., 10% of possible ratings known) to dense ones (e.g., 100% known), we assessed how each CF technique will perform with limited and abundant user-item matrices in Section 6.2. This testing provided insights into the evaluation metrics of the algorithms in handling with real-world conditions where data sparsity is usually unavoidable.

4.4.2 Collaborative Filtering Techniques Explored

In this research, we explored CF techniques using the `Surprise` Python library [16]. It offers a collection of algorithms for building and analyzing recommender systems. Before implementing the algorithms, we prepared our dataset by generating synthetic user ratings based on NLP-based CBF similarity scores. We introduced different levels of data sparsity (i.e., 15% or 0.15) by controlling the percentage of known ratings in the user-item interaction matrix.

First, we create a function for each sparsity level to randomly select a subset of user-item pairs to include in the dataset. The sparsity level determines the proportion of ratings to be kept in the dataset, simulating real-world scenarios where *users rate only a small subset of available items.* The function first calculates the average number of ratings each user is expected to provide based on the sparsity level and the total number of items (e.g., university programs).

For each user, a random subset of items is selected to be rated, ensuring that the number of ratings approximately matches the calculated threshold. These selected ratings are then extracted from the original dense ratings matrix and added to the sparse dataset. Any unselected user-item pairs are set to `NaN`, representing missing ratings. The resulting sparse matrix contains only a subset of the original user-item ratings, controlled by the specified sparsity level. This matrix serves as input for testing the performance of recommendation techniques discussed later.

4.4.2.1 K-Nearest Neighbors User-based CF

The K-Nearest Neighbors (KNN) algorithm predicts the output for a given instance by finding the K most similar instances (neighbors) in a dataset. The similarity between instances is, in this case, calculated using cosine similarity as explained in Section 4.3.3. In the context of User-based CF, it works by:

- First, present the similarity between items based on user ratings.
- Then, select the top K most similar items between users (neighbors).
- Finally, use the average users' ratings on these similar items to predict the rating for the target item.

KNN User-based CF algorithm implementation is straightforward and requires minimal tuning. In addition, the recommendation process is transparent and easy to understand, as predictions are based on observable neighbor relationships.

```
sim_options = {
    'name': 'cosine',
    'user_based': True,
    'min_support': 5,
}
user_cf = KNNBasic(k=20, min_k=1,sim_options=sim_options)
```

Figure 4.6: KNN Algorithm Implementation for User-based CF

where:

- `name`: specifies the similarity metric, with `cosine` similarity chosen.
- `user_based=True`: indicates that the CF approach is user-based.
- `min_support`: indicates that similarities are computed only when at least five common items exist between two users. Otherwise, similarity between two users does not exist.
- `k`: the number of nearest neighbors considered.
- `min_k`: at least one neighbor for a prediction to be made.

4.4.2.2 K-Nearest Neighbors Item-based CF

Item-based CF operates on principles similar to those discussed for user-based CF, with the primary distinction being that the focus shifts from identifying similarities between users to identifying similarities between items. Basically, this approach leverages the same similarity metrics and aggregation techniques but applies them to item relationships instead of user relationships. Figure 3.5 demonstrates how to implement Item-based CF with KNN Algorithm using `Surprise` Python library.

```
sim_options = {
    'name': 'cosine',
    'user_based': False,
    'min_support': 5,
}
user_cf = KNNBasic(k=20, min_k=1,sim_options=sim_options)
```

Figure 4.7: KNN Algorithm Implementation for Item-based CF

where:

- *name*: specifies the similarity metric, with cosine similarity chosen.
- *user_based=False*: indicates that the CF approach is item-based.
- *min_support*: indicates that similarities are computed only when at least five common items exist between two users. Otherwise, similarity between two users does not exist.
- *k*: the number of nearest neighbors considered.
- *min_k*: at least one neighbor for a prediction to be made.

Despite its simplistic implementation, KNN algorithms can become computationally expensive as the size of the dataset increases, particularly in high-dimensional spaces. For example, with the same parameters, in our research case, using more than 500 user preferences and 10000 program's descriptions, Item-based CF takes exponentially longer than User-based CF to compute. In theory, if the data is sparse, performance degrades as finding similar users becomes challenging, leading to reduced accuracy. This is discussed further in Section 6.2

4.4.2.3 Singular Value Decomposition

NMF is a matrix decomposition technique in model-based CF that decomposes the inputted user-item rating matrix into latent factors. SVD aims to find the underlying structure in the data by identifying patterns of user preferences and program's attributes. For example, in the context of this research, latent factors might represent topics such as "data science" or "humanities". SVD is the result of matrix multiplication of three matrices:

$$R \approx U\Sigma_{m\times n} I_{n\times n}$$

where

- R: represents the original $m \times n$ user-item matrix.
- U: represents the $m \times m$ latent factors for users.
- Σ: is a diagonal matrix containing singular values, each represents the imporance of the corresponding latent factors.
- I^T represents matrix representing the latent factors for items.

The rating predictions are computed by reconstructing the user-item matrix using the latent factors. The prediction for a user u and item i is given by:

$$r_{u,i} = U_u.\Sigma.I_i^T$$

where U_u and I_i^T are the latent factors of the user and item, respectively.

Code Implementation of SVD Using Surprise Library

```
svd_algo = SVD(n_factors=15, n_epochs=20, biased=False)
```

where:

- *n_factor*: specifies the number of latent factors to be used in the decomposition.
- *n_epochs* indicates the number of training iterations (epochs) the algorithm will perform.
- *biased*: specifies whether to use bias terms in the model. Bias terms adjust for global tendencies in the data, such as certain users giving consistently high ratings or some items receiving consistently low ratings.

4.4.2.4 Non-Negative Matrix Factorization

NMF is a matrix decomposition technique in model-based CF that decomposes the inputted user-item rating matrix into *two lower-dimensional non-negative matrices*: one representing user latent features and the other representing item latent features. These latent features capture the underlying factors that influence user preferences and item characteristics. It can be formulated as:

$$R \approx U.I$$

where

- R represents the given $m \times n$ user-item rating matrix.
- U represents a non-negative $m \times k$ matrix representing the latent factors for users, with m being the number of users and k being the number of latent factors.
- I represents a non-negative $k \times n$ matrix representing the latent factors for users, with n being the number of users and k being the number of latent factors as in U.

The rating prediction for a user preference u and item i can be computed by multiplying the approximated latent matrices:

$$r_{u,i} = U_u.I_i$$

where U_u and I_i are the latent factors for the chosen user and item, respectively.

Code Implementation of NMF Using Surprise Library

```
nmf_algo = NMF(n_factors=15, n_epochs=20, biased=False)
```

By learning latent factors that represent the underlying structure of the data, NMF and SVD are said to do better than User-based and Item-based CF in large datasets or when addressing sparsity issue. We discuss this via evaluation metrics in Section 6.2

These model-based methods are computationally intensive during the training phase, which may require significant computational resources and time. On the other hand, once trained, the model requires less memory and computational power for making predictions for making predictions for new users and items, because they do not explicitly compare users or items. Instead, they focus on uncovering deeper-level connections and latent patterns between them, enabling efficient representation and recommendation even in larger datasets.

Chapter 5

Evaluation of Synthetic User Preferences

In this chapter, we evaluate the effectiveness of synthetic user preferences in replicating real user inputs. Our primary goal is to determine how realistic the synthesized preferences are and how well they can substitute real user data in Education RecSys. In what follows, we first describe the research question and our hypothesis. Then, we explain the setup of the experiment and its artifacts. Finally, we provide the experiment's outcome and elaborate our findings.

5.1 Research question and Hypothesis

Our evaluation focuses mostly on the realism aspect. To start with, we would like to answer the following research question:

> **Research question**: Given a set of unlabeled, text-only statements designed to appear human-like, to what extent can evaluators accurately distinguish between Human- and AI-generated text?

We hypothesize that

> **Hypothesis**: Evaluators are **not** able to distinguish between Human- and AI-generated text, and their classification accuracy will not differ significantly from random chance.

This hypothesis is based on the recent advancement of artificial intelligence, in particular, large language models. As already mentioned in Chapter 2, there has been a lot of research on Generative Pre-trained Transformer across various domains, such as medicine [46], energy management [56], 3-dimension content creation [47] and cybersecurity [18]. However, this technique raises concerns about distinguishing the synthesized

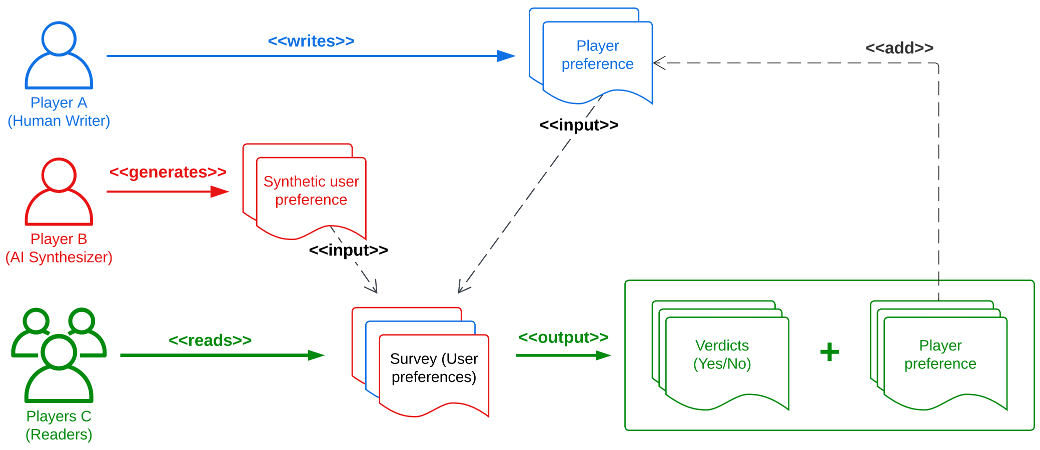


Figure 5.1: Experiment of Synthetic User Preferences Setup

outputs from human-generated text. More recently, to prevent this technology from being used, there is a new branch of research that works on tools to detect "AI-generated" artifacts and prevent misuse of this technology [11, 35].

5.2 Experiment Design

In this section, we describe our experiment in detail. Figure 5.1 describes our experiment step-by-step. In a nutshell, the experiment can be seen as our version of the imitation game-originally proposed by Alan Turing, also known as the Turing Test [51]. In this experiment, in the context of AI, there are three players:

- Player A – the Writer/Human Writer: A human-being who is tasked to write different English sentences expressing different preference of looking for master's degrees. These are authentic sentences. In this experiment, we denote this initial set of sentence is $Auth$.
- Player B – the Synthesizer/AI Writer: An AI which is tasked to synthesize different English sentences expressing different preference of looking for master's degrees. These are synthetic sentences. In this experiment, we denote this initial set of sentence is Syn.
- Player C – the Reader: A human-being who is tasked to read different English sentences expressing different preference of looking for master's degrees. These are given sentences. These are the permutation of the previous collection of sentences. We denote this set of sentence by $Given$.

For each sentence, the Reader must choose whether it is coming from Player A (i.e., written by the Writer) or from Player B (i.e., generated by the Synthesizer). If the verdict is the latter, then the Reader has the option to become the Writer himself: he can write his own sentence that is semantically equivalent with the assumed-to-be generated sentence.

We perform this game for 2 rounds, each round consists of 10 to 15 Readers (the set of readers are disjoint). At the end of each round, we collect this set of new writers' sentences; we denote it by $Auth_i$, where i is the round number (0 means it is the base set). For the sake of fairness, we limit the experiment to 20 sentences per round. Furthermore, we ensure that these sentences maintain a 1:3 ratio, with 5 contributed by Player A and 15 by Player B.

1. *Round 1*: $Auth_1 = Auth_0$, $Syn = Syn_0$,
   and $Given = Auth_1 \cup Syn$.
2. *Round 2*: $Auth_2 = Auth_0 \cup Auth_1$, $Syn = Syn_0$,
   and $Given = Auth_2 \cup Syn$.

where $\mid Auth_i \mid = 5$ and $\mid Syn \mid = 15$.

At the beginning of the experiment, we assume the role of the Writer. We selected ChatGPT model-4o-mini [31] to be the Synthesizer. For the group of Readers, we targeted engineering students, as well as young adults who are well-exposed with technology. In particular, we have 14 Readers for *Round 1*, and 10 Readers for *Round 2*.

5.3 Result and Discussion

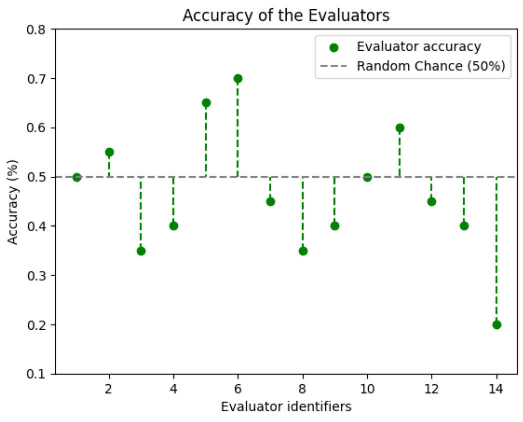
Remarks The results from the two rounds of the survey provide insights into human evaluators' ability to distinguish between AI-generated and human-written text. As random guessing would yield an accuracy of 50 %, we use this as the baseline against the performance of the human evaluators, i.e., any significant deviation from this baseline would indicate that evaluators could distinguish between the text types. Conversely, results close to 50% suggest that human evaluators struggle to identify AI-written text reliably.

5.3.1 Survey outcome

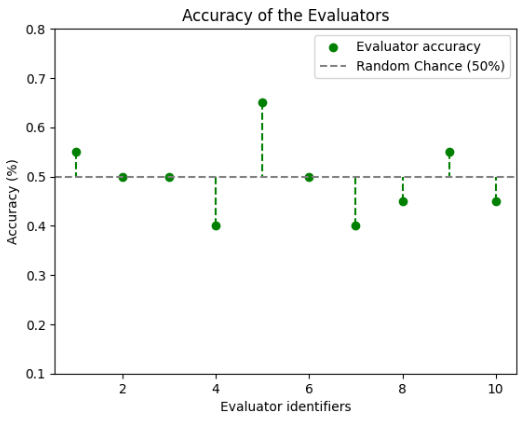
In Table 5.1, we present the overall evaluation result on the standard metrics of the survey. Individual scores are described in Figure 5.2b. In addition, the confusion matrices of evaluator predictions in *Round 1* and *Round 2* are shown in Figure 5.3a and Figure 5.3b.

	Mean	**Precision**	**Recall**	**t-Statistic (t_{stat})**	**p-Value (p_{value})**
Round 1	0.464	0.80	0.38	-1.01	0.33
Round 2	0.495	0.77	0.47	-0.21	0.84
Total	0.477	0.785	0.42	-1.01	0.32

Table 5.1: Survey evaluation results

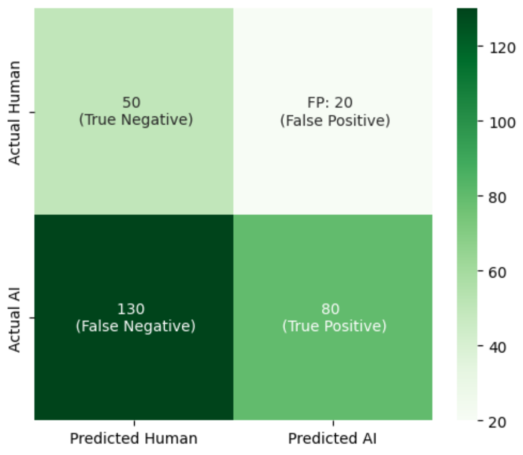


(a) *Round 1*

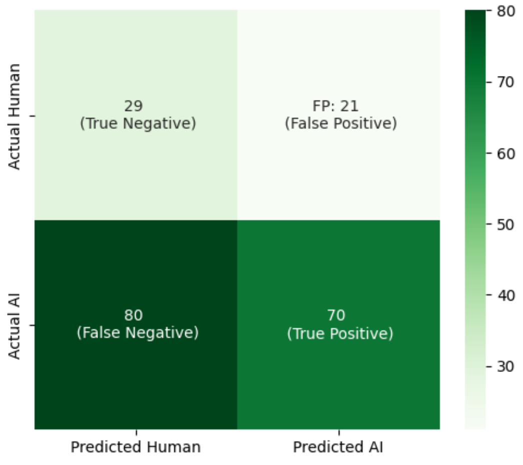


(b) *Round 2*

Figure 5.2: Distribution of players accuracy, against the baseline of 50%.



(a) *Round 1*



(b) *Round 2*

Figure 5.3: Confusion matrix of evaluator predictions

5.3.2 Discussion

In general, the result obtained from the survey provides strong evidence that supports our hypothesis, i.e., evaluators are **not** able to distinguish between Human- and AI-generated text, and their classification accuracy will not differ significantly from random chance.

In *Round 1*, the average accuracy across 14 participants was 46.43%, while it is 49.5% over the 10 participants in the second round. These *mean accuracy* result is close to random chance (50%). This supports the hypothesis that evaluators are unable to reliably distinguish AI-generated text from human-written text, performing closely to random guessing.

Precision values were relatively high in both rounds, with 0.80 in *Round 1* and 0.77 in *Round 2*. This indicates that when evaluators predicted a statement as AI-written, they were often correct. High precision suggests that evaluators were conservative in their predictions, meaning they focus on ensuring that their AI-written classifications were accurate. However, the high precision observed can be explained by the imbalance in the dataset, where the *ratio of AI- and Human-generated statements is 3:1*. This imbalance skews the results in favor of precision, making it an unreliable indicator of the experiment's performance.

Furthermore, a recall of 0.38 in *Round 1* (as well as 0.47 in *Round 2*, respectively) indicates that only 38% (47%) of the actual AI-written texts were correctly identified by the human evaluators.

For AI-generated text, evaluators correctly classified 80 samples as AI-written in *Round 1* and 70 samples in *Round 2*. However, the number of AI-generated samples misclassified as human-written was higher in both rounds: 130 in *Round 1* and 80 in *Round 2*. This results in recall values being low, highlighting limitations in evaluators' ability to identify AI-generated text.

Chapter 6

Evaluation

The following chapter presents the evaluation of the recommendation methodologies developed and discussed extensively in this study. The evaluation is divided into two sections:

- Qualitative assessment of NLP-based CBF Algorithm in Section 6.1
- Quantitative assessment using MSE (Root Mean Squared Error) and MAE (Mean Absolute Error) metrics for four Collaborative Filtering (CF) techniques at different data sparsity levels (0%, 25%, 70%, 90%) in Section 6.2.

6.1 Qualitative assessment of NLP-based Content-based Filtering

The qualitative assessment of recommendations generated by NLP-based CBF can be referred to as an "Eye Test". Basically, this experiment involves manually inspecting programs' recommendations for a set of users to determine if the suggestions are relevant and make *intuitive* sense based on the users' preferences.

Results and Observations

We sampled 5 users from the original user dataset. These users are designed to represent a diverse range of profiles.

User 1:

- Preference Statement: “A Master of Science in Energy Engineering in Stuttgart would be an excellent choice, focusing on renewable and efficient energy systems.”
- Top Recommendations:

Ranking	Program Details	Degree Type
1	Sustainable Energy Competence (SENCE) Stuttgart University of Applied Sciences Stuttgart, Germany	Master
2	Green Energy Westcoast University of Applied Sciences Heide, Germany	Master of Science
3	Renewable Energy Systems Hochschule Nordhausen Nordhausen, Germany	Master

- Observation: The first option is the best match for this user preference, as it aligns with the user’s focus, degree preference, and location. The other two programs are relevant in terms of focus but fail to meet the location preference, making them less ideal choices, but they could still be considered strong alternatives

User 2:

- Preference Statement: “I’d like to pursue a Master of Arts in Theatre Direction in Leipzig, where I can develop my skills in stage management and storytelling.”
- Top Recommendations:

Ranking	Program Details	Degree Type
1	Stage and Film Make-up Munich College of Music and Theatre München, Germany	Master of Arts
2	Film, Television and Digital Narratives SRH University of Applied Sciences Heidelberg Berlin, Germany	Master of Arts
3	Cultures of Curators College of Graphic and Book Arts, Leipzig Leipzig, Germany	Master’s

- Observation: While the recommendations provide relevant programs, none perfectly aligns with all of the user’s stated preferences. The first two programs meet the user’s preference, focusing on stage management and threatre direction. As opposed to the last option, which does not align completely with the user’s interest but matches the desired location, which could be a strong advantage.

User 3:

- Preference Statement: " An English-taught Master's in logistics and supply chain management interests me."
- Top Recommendations:

Ranking	Program Details	Degree Type
1	Global Logistics and Supply Chain Management Kühne University for Logistics and Company Management Hamburg, Germany	Master of Science
2	Supply Chain and Operations Management Technische Hochschule Köln Köln, Germany	Master
3	Sustainable Supply Chain Management SRH University of Applied Sciences Heidelberg Berlin, Germany	Master of Science

- Observation: all of three option are great recommendations due to its strong alignment with the user's preferences.

User 4:

- Preference Statement: "A Master's in advanced aeronautics would align with my passion for space exploration."
- Top Recommendations:

Ranking	Program Details	Degree Type
1	Planetary Sciences and Space Exploration Free University Berlin Berlin, Germany	Master
2	Space Engineering Berlin University of Technology Berlin, Germany	Master
3	ESPACE - Earth Oriented Space Science and Technology Munich University of Technology München, Germany	Master of Science

- Observation: All three options prove to be solidly align with user's preference for advanced aeronautics.

User 5:

- Preference Statement: "I'm interested in degrees in public policy and governance, particularly those with a focus on equity."
- Top Recommendations:

Ranking	Program Details	Degree Type
1	Policy Management, Public Policy and Administration University of Duisburg-Essen Duisburg, Germany	Master of Arts
2	Public Policy University of Erfurt Erfurt, Germany	Master of Public Policy
3	Master of Public Policy University of Duisburg-Essen Duisburg, Germany	Master of Public Policy

- Observation: None of the programs explicitly mention a focus on equity, which suggests a need for either deeper exploration of program dataset. However, these recommendations suggest adequate interpretation into the user's preferences.

6.2 Quantitative evaluation for Collaborative Filtering Techniques

In this subsection, we quantitatively evaluate four Collaborative techniques using RMSE and MAE metrics across different data sparsity levels. The CF techniques analyzed are: (i) KNN User-based CF, (ii) KNN Item-based CF, (iii) NMF, and (iv) SVD.

Experiment Design

Synthetic Dataset is generated by mapping NLP-based CBF similarity scores to a standardized rating scale (1 to 5) with added noise to simulate variability. The evaluation focuses on the performance of the algorithms under varying levels of data sparsity, with dataset densities set at 0%, 25%, 70%, and 99%. This allows an analysis of how each algorithm performs under various degrees of sparsity, a common challenge in recommendation systems.

To ensure robust results, the experiment employs cross-validation, splitting the data into training and testing sets multiple times to assess the consistency and generalization of the algorithms. The quality of recommendations is evaluated using two widely recognized metrics: RMSE (Root Mean Squared Error), which measures the average squared difference between predicted and actual ratings and MAE, which calculates the average prediction errors.

Results

The following tables present the RMSE and MAE for each CF technique at different sparsity levels.

Algorithm	RMSE	MAE
User-based CF	0.6791	0.5335
Item-based CF	0.7050	0.5631
NMF	0.5904	0.4726
SVD	0.4956	0.4001

(a) at 0% Data Sparsity

Algorithm	RMSE	MAE
User-based CF	0.6769	0.5336
Item-based CF	0.7034	0.5596
NMF	0.5920	0.4738
SVD	0.4982	0.4020

(b) at 25% Data Sparsity

Algorithm	RMSE	MAE
User-based CF	0.6832	0.5390
Item-based CF	0.7060	0.5610
NMF	0.5994	0.4793
SVD	0.5155	0.4153

(c) at 70% Data Sparsity

Algorithm	RMSE	MAE
User-based CF	0.7653	0.6130
Item-based CF	0.7220	0.5749
NMF	0.7631	0.6109
SVD	0.6692	0.5383

(d) at 99% Data Sparsity

Table 6.1: RMSE and MAE for Different Algorithms across Data Sparsities

Analysis

By intentionally training on a sparse subset of our data and testing on the full dataset, we are examining how the model handles situations where many users and items are unknown—a common real-world cold-start problem. This setup allows us to measure the effect of sparsity by observing the evaluation metrics (i.e., RMSE, MAE) as sparsity increases provides quantitative evidence of how data sparsity affects model performance.

Though the results indicate that SVD and NMF generally outperform User-Based CF and Item-Based CF at higher sparsity levels, they do not definitively establish their superiority across all contexts. The observed performance differences are contingent on various factors, including dataset characteristics, sparsity levels, and the evaluation metrics used. While SVD and NMF demonstrate strength, performing well in a relatively large dataset, their effectiveness diminishes significantly under extreme sparsity conditions.

However, it does confirm that increasing sparsity leads to higher RMSE and MAE values, indicating a decline in prediction accuracy, this observation highlights the inherent challenges of recommendation systems in handling sparse datasets. The results emphasize the importance of addressing sparsity to improve the reliability of recommendations, but further experimentation is needed to explore methods for mitigating its impact across different algorithms and conditions.

Chapter 7

Conclusion

This thesis explores the development and evaluation of a number of recommender systems for university programs, leveraging different techniques from both CBF and CF. Through the integration of NLP-based CBF, synthetic user data generation, similarity scoring computations, this research introduces recommendations according to the variability in user preferences. Furthermore, with the utilization of various CF techniques, both Memory-based and Model-based, the study addresses the key challenge in recommendation methodologies, which is data sparsity.

The research begins with the simulation of synthetic user preference dataset generated using OpenAI's ChatGPT. Combining with the data extracted from the German university portal, they serve as the foundation for developing NLP-based CBF algorithm, which utilizes `TF-IDF` and `Word2Vec` to extract meaningful features from program descriptions and user preferences. Weighted sentence embeddings, constructed by combining Word2Vec vectors with TF-IDF weights, are used to compute cosine similarity scores. These scores are ranked and evaluated for the recommendation process. The results written in Section 6.1 demonstrate that this approach effectively personalizes recommendations by aligning program attributes with user preferences, even in the absence of explicit ratings.

In addition to CBF, the study investigates Collaborative Filtering techniques, including User-Based CF, Item-Based CF, SVD , and NMF. The results reveal that SVD perform best in scenarios with large dataset and higher data sparsity. However, the study also shows that increased sparsity negatively impacts RMSE and MAE for all algorithms, highlighting the critical challenge of sparsity in recommendation systems.

Despite these findings, the study acknowledges certain limitations. While the results suggest that some algorithms perform better than others under specific conditions, the conclusions are not universally applicable. Factors such as dataset characteristics, user variability, and experimental settings influence the outcomes, necessitating further testing on diverse datasets to validate the generalizability of the results. Also, synthetic user which is proved to be capable to replicate partially real users, they are not necessarily bias-free and may not fully represent real-world user behavior.

Contributions

This research effectively contributes to the field of recommender systems by:

- Conducting a comprehensive and thorough literature review of common recommendation techniques, in theory and in practice, specifically in academic domains.
- Utilizing ChatGPT model of OpenAI to generate, to the best of our knowledge, the first synthetic database of users' preferences, addressing real-world lack of student' preferences database.
- Evaluating relevance and effectiveness of NLP-based CBF through the integration of `TF-IDF` and `Word2Vec` for feature extraction and cosine similarity for recommendation generation.
- Understanding through accuracy metrics such as RMSE and MAE how data sparsity affect the performance of different CBF techniques.

Future Work

For future work, several avenues are promising in recommender systems:

- Exploring hybrid RecSys models that combine the strengths of CBF and CF to improve recommendation accuracy in sparse datasets. For instance, when user interaction data is limited, the system can rely on item attributes (CBF). This exploration could focus on identifying optimal ways to combine these techniques, such as *weighted hybrids* or *switching models.*
- Investigating Knowledge-based RecSys that use constraints to better evaluate and generate recommendations that best suit users. This investigation will focus on developing *constraint-based models* to evaluate user requirements more effectively and generate highly relevant recommendations
- The performance of recommendation systems, in this research, is validated on synthetic datasets, which may not fully reflect real-world complexities. Extending the evaluation to real-world datasets provides a more accurate assessment of the system's applicability.
- Advanced neural network models, such as deep learning architectures, offer significant potential for enhancing recommendation systems by capturing complex patterns in user preferences and program characteristics. This could explore how these neural techniques can complement traditional methods to improve recommendation accuracy, and the system's ability to generalize across datasets.

Bibliography

[1] Ashrf Althbiti, Rayan Alshamrani, Tami Alghamdi, Stephen Lee, and Xiaogang Ma. Addressing Data Sparsity in Collaborative Filtering Based Recommender Systems Using Clustering and Artificial Neural Network. In *11th IEEE Annual Computing and Communication Workshop and Conference, CCWC 2021, Las Vegas, NV, USA, January 27-30, 2021*, pages 218–227. IEEE, 2021.

[2] Minh Vinh Nguyen Phuoc Bao. GitHub Repository containig the entire process of this research paper, November 2024. https://github.com/minhnguyen1312/recommender-systems-masters.

[3] Melania Berbatova. Overview on NLP Techniques for Content-based Recommender Systems for Books. In Venelin Kovatchev, Irina P. Temnikova, Branislava Sandrih, and Ivelina Nikolova, editors, *Proceedings of the Student Research Workshop Associated with RANLP 2019, Varna, Bulgaria, September 2-4, 2019*, pages 55–61. INCOMA Ltd., 2019.

[4] Greta Björklund, Magdalena Bohlin, Edvard Olander, Josef Jansson, Cicero Eduardo Walter, and Manuel Au-Yong-Oliveira. An exploratory study on the spotify recommender system. In Álvaro Rocha, Hojjat Adeli, Gintautas Dzemyda, and Fernando Moreira, editors, *Information Systems and Technologies - WorldCIST 2022, Volume 2, Budva, Montenegro, 12-14 April, 2022*, volume 469 of *Lecture Notes in Networks and Systems*, pages 366–378. Springer, 2022.

[5] Robin D. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Model. User Adapt. Interact.*, 12(4):331–370, 2002.

[6] Aqil Burney, Muhammad Asif, Zain Abbas, and s.M.Aqil Burney. Google Maps Security Concerns. *Journal of Computer and Communications*, 06:275–283, January 2018.

[7] Sang-Min Choi, Dongwoo Lee, Kiyoung Jang, Chihyun Park, and Suwon Lee. Improving Data Sparsity in Recommender Systems Using Matrix Regeneration with Item Features. *Mathematics*, 11(2), 2023.

[8] Isaac Chuang and Andrew D. Ho. HarvardX and MITx: Four years of open online courses: Fall 2012 to Summer 2016. Technical Report 10, HarvardX Working Paper, 2016.

[9] Georgina Curto, Mario Acosta, Flavio Comim, and Begoña Garcia-Zapirain. Are AI systems biased against the poor? A machine learning analysis using Word2Vec and GloVe embeddings. *AI & SOCIETY*, 39, June 2022.

[10] Surabhi Dwivedi and V S Kumari Roshni. Recommender system for big data in education. In *2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH)*, pages 1–4, 2017.

[11] Ahmed Elkhatat, Khaled Elsaid, and Saeed Almeer. Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text. *International Journal for Educational Integrity*, 19, September 2023.

[12] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using Collaborative Filtering to weave an Information Tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[13] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), December 2016.

[14] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[15] Der Hochschulkompass. Hochschulkompass – Studieren und promovieren in Deutschland. https://hochschulkompass.de/. Accessed: 2024-11-22.

[16] Nicolas Hug. Surprise: A Python library for recommender systems, 2017. GitHub repository.

[17] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction.* Cambridge University Press, January 2010.

[18] Danial Khosh Kholgh and Panos Kostakos. PAC-GPT: A Novel Approach to Generating Synthetic Network Traffic With GPT-3. *IEEE Access*, 11:114936–114951, 2023.

[19] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 233–240. ACM, 2016.

[20] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[21] Ashok Koujalagi. Determine Word Relevance in Document Queries Using TF-IDF. *International journal of scientific research*, 4, 2015.

[22] Saurabh Kulkarni and Sunil F. Rodd. Context Aware Recommendation Systems: A review of the state of the art techniques. *Comput. Sci. Rev.*, 37:100255, 2020.

[23] Adam Lesnikowski, Gabriel de Souza Pereira Moreira, Sara Rabhi, and Karl Byleen-Higley. Synthetic Data and Simulators for Recommendation Systems: Current State and Future Directions, 2021.

[24] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Syst. Appl.*, 41(4):2065–2073, 2014.

[25] Chaimaa Lotfi, Swetha Srinivasan, Myriam Ertz, and Imen Latrous. *Web Scraping Techniques and Applications: A Literature Review*, pages 381–394. SCRS Conference Proceedings, January 2021.

[26] Jakub Macina, Ivan Srba, Joseph Jay Williams, and Maria Bielikova. Educational Question Routing in Online Student Communities. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 47–55, New York, NY, USA, 2017. Association for Computing Machinery.

[27] Chahrazed Mediani, Saad Harous, and Mahieddine Djoudi. Content-Based Recommender System using Word Embeddings for Pedagogical Resources. In *2023 5th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–8, 2023.

[28] Spotify's Moderator. Introducing to the Spotify Smart Shuffle!, March 2023. https://community.spotify.com/t5/Your-Library/Introducing-Smart-Shuffle/td-p/5516566/page/43.

[29] Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings*, volume 9626 of *Lecture Notes in Computer Science*, pages 729–734. Springer, 2016.

[30] Charbel Obeid, Inaya Lahoud, Hicham El Khoury, and Pierre-Antoine Champin. Ontology-based recommender system in higher education. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 1031–1034. ACM, 2018.

[31] OpenAI. GPT-4o mini: advancing cost-efficient intelligence, 2024.

[32] OpenAI. ChatGPT. https://openai.com/index/chatgpt/, n.d. Accessed: 2024-11-10.

[33] Laura Pappano. The Year of the MOOC. *The New York Times*, 2012. Retrieved from https://www.nytimes.com.

[34] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, Montreal, QC, Canada, October 17-19, 2016*, pages 399–410. IEEE, 2016.

[35] Mike Perkins, Jasper Roe, Darius Postma, James McGaughran, and Don Hickerson. Detection of GPT-4 Generated Text in Higher Education: Combining Academic Judgement and Software to Identify Generative AI Tool Misuse. *Journal of Academic Ethics*, 22:1–25, October 2023.

[36] Selenium Project. SeleniumHQ Browser Automation. https://www.selenium.dev/, 2004–2024. Accessed: 2024-11-22.

[37] Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. *n.d.*, January 2003.

[38] GroupLens Social Computing Research. MovieLens Database - Ratings Data For Movies. https://license.umn.edu/product/movielens-database.

[39] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems: Techniques, Applications, and Challenges*, pages 1–35. Springer US, New York, NY, 2022.

[40] Elaine Rich. User modeling via stereotypes. *Cogn. Sci.*, 3(4):329–354, 1979.

[41] Leonard Richardson. Beautiful Soup Documentation. https://www.crummy.com/software/BeautifulSoup/, 2004–2024. Accessed: 2024-11-22.

[42] Bryan Robinson. Six Tips To Prevent Decision Fatigue From Short-Circuiting Your Mind And Career, July 2024.

[43] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on Recommender Systems. *Journal of Big Data*, 9, 2022.

[44] Francesco Scarcello. Artificial Intelligence. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 287–293. Academic Press, Oxford, 2019.

[45] Barry Schwartz. *The Paradox of Choice*, chapter 8, pages 121–138. John Wiley & Sons, Ltd, 2015.

[46] Fahim Sufi. Addressing Data Scarcity in the Medical Domain: A GPT-Based Approach for Synthetic Data Generation and Feature Extraction. *Information*, 15(5), 2024.

[47] Zeyi Sun, Tong Wu, Pan Zhang, Yuhang Zang, Xiaoyi Dong, Yuanjun Xiong, Dahua Lin, and Jiaqi Wang. Bootstrap3D: Improving Multi-view Diffusion Model with Synthetic Data, 2024.

[48] Thoufeeq Ahmed Syed, Vasile Palade, Rahat Iqbal, and Smitha Sunil Kumaran Nair. A Personalized Learning Recommendation System Architecture for Learning Management System. In Ana L. N. Fred and Joaquim Filipe, editors, *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - (Volume 1), Funchal, Madeira, Portugal, November 1-3, 2017*, pages 275–282. SciTePress, 2017.

[49] The Guardian. Fitness tracking app gives away location of secret US army bases, January 2018. Accessed: 2024-11-17.

[50] The Washington Post. Elon Musk's jet tracker highlights tensions between free speech and doxxing, December 2022. Accessed: 2024-11-17.

[51] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, October 1950.

[52] Thirunavu Va, Vishwa Ravi, Rojith Murugan, and Tamilkumaran. Movie Recommendation System, February 2024.

[53] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2643–2651, 2013.

[54] Jing Wang and Jian Yin. Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity. In Jean X. Gao, Dongrong Xu, Xiaoyan Sun, and Yingfei Wu, editors, *6th International Conference on Biomedical Engineering and Informatics, BMEI 2013, Hangzhou, China, December 16-18, 2013*, pages 661–665. IEEE, 2013.

[55] Dianshuang Wu, Jie Lu, and Guangquan Zhang. A Fuzzy Tree Matching-Based Personalized E-Learning Recommender System. *IEEE Trans. Fuzzy Syst.*, 23(6):2412–2426, 2015.

[56] Chaobo Zhang, Jie Lu, and Yang Zhao. Generative pre-trained transformers (GPT)-based automated data mining for building energy management: Advantages, limitations and the future. *Energy and Built Environment*, 5:143–169, June 2023.

[57] Radim Řehůřek and Petr Sojka. Gensim: Topic Modelling for Humans. `https://radimrehurek.com/gensim/`, 2010. Accessed: 2024-11-22.

Appendix

A Synthetic data: Master's Program Preferences

No.	Program Description
1	I am looking for a two-year full-time master's program in Data Science, focusing on artificial intelligence and machine learning, taught in English, with an emphasis on practical experience and internship opportunities.
2	I prefer a one-year online master's program in Business Administration that specializes in entrepreneurship, ideally offered in English, with options for real-world project work.
3	My ideal master's program would be a two-year full-time course in Renewable Energy Engineering, focused on sustainable energy practices, taught in English, and include hands-on lab experience.
4	I seek a part-time master's degree in Educational Technology, lasting two years, with a specialization in digital learning tools, taught in English, and offering a flexible schedule for working professionals.
5	I'm interested in a one-year full-time master's program in Public Health that emphasizes global health and policy, preferred in English, with strong internship placements.
6	I would like to pursue a two-year master's in International Relations, focusing on conflict resolution, taught in English, with opportunities for study abroad and diplomatic internships.
7	I am aiming for a two-year online master's in Cybersecurity, with a specialization in network security, instructed in English, and including a capstone project that simulates real-world challenges.
8	I desire a one-year full-time master's program in Creative Writing focusing on fiction, taught in English, with ample opportunities for workshops and feedback from industry professionals.
9	I am looking for a two-year part-time master's in Nursing with a concentration in psychiatric care, offered in English, combining online courses and clinical experiences.
10	I want to enroll in a one-year full-time master's in Environmental Science, specializing in climate change impacts, taught in English, with fieldwork opportunities.
11	My preferred program is a two-year full-time master's in Marketing, focusing on digital marketing strategies, taught in English, with an internship component included.
12	I would like to undertake a two-year online master's in Applied Mathematics with a specialization in statistics, taught in English, with access to virtual labs and collaboration tools.
13	I am interested in a one-year full-time master's in Graphic Design that emphasizes user experience, offered in English, with extensive portfolio-building opportunities.
14	I prefer a two-year part-time master's in Finance, focusing on investment analysis, conducted in English, with the possibility of networking events and workshops.
15	I want to pursue a full-time, one-year master's in Social Work specializing in community practice, taught in English, with practical field placements included.
16	I seek a two-year master's degree in Human Resources Management, focusing on organizational development, offered in English, combining virtual learning with internships.
17	I'm looking for a one-year full-time master's in Artificial Intelligence, emphasizing natural language processing, with a program taught in English and an applied research project.
18	I want to enroll in a two-year part-time master's in Supply Chain Management, with a focus on logistics optimization, taught in English, and featuring collaboration with industry partners.
19	I am aiming for a one-year full-time master's in Fashion Design, specializing in sustainable fashion, conducted in English, with regular industry mentorship.
20	I seek a two-year online master's in Software Engineering, focusing on cloud computing technologies, instructed in English, and with a rigorous coding bootcamp component.
21	I prefer a one-year master's in Political Science, specializing in environmental policy, taught in English, with field research opportunities.
22	I would like to pursue a two-year full-time master's in History, focused on cultural heritage and conservation, conducted in English, with hands-on projects in museums.
23	I'm interested in a one-year part-time master's in Occupational Therapy, focusing on rehabilitation techniques, taught in English, with requirements for practical experience.

(Continue next page)

No.	Program Description
24	I want to enroll in a two-year full-time master's in Agri-Food Systems, with a specialization in sustainable agriculture, conducted in English, and including internships with local farms.
25	I am looking for a one-year full-time master's in Film Studies, with a focus on documentary film-making, taught in English, that includes workshops and film production projects.
26	I seek a two-year part-time master's in Information Technology Management, specializing in data analytics, taught in English, with project-oriented learning and case studies.
27	I would like to pursue a one-year online master's in Bioinformatics, with a specialization in genomic data analysis, offered in English, and featuring access to leading research labs.
28	My ideal program is a two-year full-time master's in Real Estate Development, focusing on urban planning and sustainability, taught in English, with exposure to industry practices.
29	I am interested in a one-year full-time master's in Music Production, specialized in audio engineer-ing, conducted in English, with studio access for practical learning.
30	I want to enroll in a two-year part-time master's in Data Analytics, specializing in big data tech-nologies, instructed in English, and including networking events with tech companies.
31	I seek a one-year full-time master's in Performance Psychology with a focus on sports, taught in English, and offering hands-on coaching experiences.
32	I would like to pursue a two-year online master's in Blockchain Technology, specializing in cryp-tocurrency applications, taught in English, with industry-aligned case studies.
33	I am looking for a one-year full-time master's in Journalism, with a focus on investigative report-ing, offered in English, with practical projects in media.
34	I prefer a two-year part-time master's in Electrical Engineering, specializing in renewable energy systems, taught in English, with capstone project requirements.
35	I desire a one-year full-time master's in Animation, with a focus on 3D modeling, taught in English, featuring collaborations with studios for hands-on experience.
36	I seek a two-year full-time master's in International Business, specializing in cross-cultural man-agement, conducted in English, with exchanges available with partner universities.
37	I want to enroll in a one-year online master's in Disaster Management, with a focus on crisis re-sponse strategies, offered in English, featuring simulation exercises.
38	My ideal program is a two-year part-time master's in Data Science, focusing on data visualization techniques, taught in English, with strong project collaboration opportunities.
39	I am interested in a one-year full-time master's in Biomedical Engineering, specializing in medical device design, instructed in English, with industry partnerships for design challenges.
40	I seek a two-year online master's in Human-Computer Interaction, focusing on usability research, taught in English, with collaborative projects with industry professionals.
41	I would like to pursue a one-year full-time master's in Theology, specializing in social justice issues, taught in English, with community engagement components.
42	I want to enroll in a two-year part-time master's in Graphic Communication, focusing on branding and identity design, instructed in English, with real-world client projects.
43	I am aiming for a one-year full-time master's in Digital Marketing, emphasizing content strategy, taught in English, with an option for practical workshops.
44	My preferred program is a two-year full-time master's in Sports Management, specializing in event management, offered in English, with hands-on industry training.
45	I seek a one-year master's in Urban Planning, with a focus on sustainable cities, conducted in English, including fieldwork on local urban projects.
46	I want to pursue a two-year part-time master's in Hospitality Management, with a specialization in sustainable tourism, taught in English, featuring industry internships.
47	I am looking for a one-year full-time master's in Entrepreneurial Leadership, focusing on startup development, instructed in English, with mentoring from successful entrepreneurs.
48	I desire a two-year online master's in Supply Chain Analytics, focusing on risk management, taught in English, and including case studies from industry leaders.
49	I would like to enroll in a one-year full-time master's in Family Therapy, specializing in counseling techniques, offered in English, with mandatory clinical placements.
50	I am interested in a two-year part-time master's in Statistics, focusing on biostatistics applications, directed in English, with opportunities for research collaborations.

Table A.1: Master's Program Synthetic User Preferences (🟨 highlights the specific preferences)