Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Ingrid Amalie Solbjørg

Using Sentiment Analysis to Improve Course Recommendations for MOOCs

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2023

NTNU
Norwegian University of
Science and Technology

Ingrid Amalie Solbjørg

Using Sentiment Analysis to Improve Course Recommendations for MOOCs

Master's thesis in Computer Science
Supervisor: Özlem Özgöbek
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Massive Open Online Courses (MOOCs) have been used in e-learning over the last decade, and their emergence sparked during the COVID-19 pandemic. New courses are becoming available frequently, making the learners overwhelmed and struggling to find courses that align with their interests. Combined with the drop-out rate on MOOCs being 90%, this makes it even more critical for learners to find suitable courses. As a result, Recommender Systems have emerged to reduce the time spent searching by filtering out irrelevant courses and recommending the most relevant ones to learners. However, these systems must be improved to help learners find suitable courses to reduce the drop-out rate in MOOCs.

Recent works in other domains show that combining reviews with ratings improves recommendations. This thesis aims to adopt this idea by developing a Recommender System that incorporates sentiment from course reviews into Course Recommendations. The sentiment is extracted by performing Sentiment Analysis using a BERT model and combined with the original ratings using weights. A set of recommendation algorithms were implemented to analyze the impact of the adjusted ratings. Then, the Recommender System was evaluated on the COCO dataset, containing 4.5M course reviews.

Ultimately, all the recommendation algorithms performed slightly better with the adjusted ratings. The algorithm with the most significant improvement was *ALSImplicitMF*, which improved its nDCG score by 1.54%. However, the overall performances of the algorithms were poor compared to related research, partly because of the sparsity of the dataset.

Sammendrag

Massive åpne nettkurs (MOOCer) har blitt brukt i e-læring det siste tiåret, og fremveksten deres eksploderte under COVID-19-pandemien. Nye kurs blir stadig tilgjengeliggjort, noe som gjør at studentene blir overveldet og sliter med å finne kurs som passer deres interesser. Dette, kombinert med at frafallet på MOOCer er 90%, gjør det enda viktigere for studentene å finne passende kurs. Som et resultat har anbefalingssystemer blitt utviklet for å redusere tiden som brukes på å finne kurs, ved å filtrere ut irrelevante kurs og anbefale de mest relevante til studentene. Disse systemene må imidlertid forbedres for å hjelpe studentene med å finne passende kurs og redusere frafallet fra MOOCer.

Nylig forskning innen andre fagområder viser at det å kombinere anmeldelser med tallvurderinger forbedrer anbefalinger. Målet med denne oppgaven er å adoptere denne ideen ved å utvikle et anbefalingssystem som inkorporerer sentiment fra kursanmeldelser i kursenes tallvurderinger. Sentimentene uthentes gjennom sentimentanalyse ved hjelp av en BERT-modell, og kombineres deretter med de opprinnelige tallvurderingene ved bruk av vekter. En rekke anbefalingsalgoritmer ble implementert for å analysere innvirkningen av de justerte tallvurderingene. Deretter ble anbefalingssystemet evaluert på COCO-datasettet, som inneholder 4,5 millioner kursanmeldelser.

Alle anbefalingsalgoritmene presterte noe bedre med de justerte tallvurderingene. Algoritmen med den største forbedringen var *ALSImplicitMF*, som forbedret sin nDCG-score med 1,54%. Imidlertid var algoritmene generelt dårlige sammenlignet med lignenede forskning, blant annet siden datasettet har få interaksjoner per student og kurs.

Preface

Before you lies the master thesis "Using Sentiment Analysis to Improve Course Recommendations for MOOCs". This thesis was written at the Norwegian University of Science and Technology (NTNU) for the Department of Computer Science (IDI) during the spring of 2023.

I was introduced to recommender systems during my exchange year in Madrid in my previous school year. This sparked my interest in the subject and made me contact my current supervisor, Associate Professor Özlem Özgöbek at NTNU, to write my thesis in the field of recommender systems. Even though I only had some basic knowledge about the topic, she let me in on one of her projects, which I am grateful for today. I want to thank Özlem for her valuable guidance and ideas, which lead us to fruitful discussions.

This work would not have been possible without a dataset. I want to express my appreciation to Dr. Mirko Marras and the owners of the COCO dataset for letting me use their dataset to conduct my research. Additionally, I want to thank NTNU and the team hosting the IDUN cluster for letting me run code that required extensive computational power.

Lastly, I would like to thank my family and friends for their support throughout the work of my thesis and during the years of my studies.

Contents

Figures

Tables

Acronyms

AI Artificial Intelligence. 13, 15

ALS Alternating Least Square. 7, 49, 65

AP Average Precision. 12

BERT Bidirectional Encoder Representations from Transformers. iii, v, 15, 22, 39, 40, 46, 47, 50, 63, 66, 70, 73

CBF Content-Based Filtering. 7–9, 19, 22, 41, 67

CF Collaborative Filtering. 6–9, 19–22, 42, 65, 67

COCO Collection of Online Courses. iii, v, vii, 25–30, 32, 33, 37, 39, 40, 42, 43, 46, 48–51, 53, 66, 69–71, 73–75, 91

CR Course Recommendation. iii, 2, 3, 12, 13, 19–21, 23, 39, 73

CS Course Selection. 4, 17–19

DCG Discounted Cumulative Gain. 12, 39

GPA Grade Point Average. 17, 19

GPT Generative Pre-trained Transformer. 15, 39, 40

IDCG Ideal Discounted Cumulative Gain. 12

k-NN k-Nearest Neighbors. xv, 6, 42, 50, 53, 54, 61, 65

KG Knowledge Graph. 18, 20, 21

LM Language Model. 13, 15, 16, 33, 39, 40, 66, 70, 71, 74

MAP Mean Average Precision. 12, 39, 48, 50, 53, 55, 57, 59, 62, 64, 66, 69

MF Matrix Factorization. 7, 22, 23, 42, 49

ML Machine Learning. 16, 25, 48

MOOC Massive Open Online Course. iii, v, 1–3, 12, 13, 17, 18, 20, 21, 39, 41, 42, 64, 66, 74

MRR Mean Reciprocal Rank. 39, 69

nDCG Normalized Discounted Cumulative Gain. iii, v, 11, 12, 39, 48, 50, 53, 55, 57–59, 62, 64–66, 69, 74

NLP Natural Language Processing. 3, 5, 13–16, 33

NTNU Norwegian University of Science and Technology. vii, 19

OULAD Open University Learning Analytics Dataset. 25

RS Recommender System. iii, xv, 1–13, 16–19, 21–25, 35, 37, 39, 41–43, 46, 47, 49–51, 53–71, 73–75

SA Sentiment Analysis. iii, 2, 13, 16, 21–23, 33, 39, 40, 46–48, 50, 56, 64, 66, 69–71, 73–75

SiEBERT Sentiment in English Bidirectional Encoder Representations from Transformers. 40

Glossary

Deep Learning A machine learning technique that uses artificial neural networks with multiple layers to learn hierarchical representations of data and solve complex tasks. 42

EdNet A Large-Scale Hierarchical Dataset in Education. 25

GDPR The General Data Protection Regulation is a data protection law that regulates the collection, processing, and storage of personal data. 19

LensKit A set of Python tools for experimenting with and studying recommender systems. xiii, 49, 50, 53, 54, 61, 62, 69, 70

MAE Mean Absolute Error measures the average absolute difference between the actual and predicted ratings. 69

SVD A mathematical technique used to factorize a matrix into three matrices representing the user and item latent factors and the singular values. 42

Web A global network of interconnected documents and resources accessible via the internet. Also known as the World Wide Web. 1, 2, 13, 19, 32, 73

Chapter 1

Introduction

This chapter provides an introduction to the thesis. First, the motivation of the thesis is presented in Section 1.1. Based on the thesis' motivation, the research goals and questions were formed. These are provided in Section 1.2. Finally, the contributions are summarized in Section 1.3 before an outline of the thesis is given in Section 1.4.

1.1 Motivation

The vast amount of data on the Web increases continuously. In fact, according to Marr [1], 2.5 quintillion bytes of data are created daily. Further, with the arrival of affordable technology and internet access, e-learning fully emerged during the COVID-19 pandemic. Also, during the pandemic, many people obtained technological skills leading to more people using and generating even more content on the web. This has caused an information overload problem for the users. As a result, Recommender Systems (RSs) have emerged to reduce the time spent searching through content [2]. They assist users by filtering out irrelevant information and providing the most relevant content to them.

RSs have found applications in various domains such as movies, online shopping, traveling, news, social media, and more. In the past decade, the use of RSs has also extended to education, where students in educational institutions and users of Massive Open Online Course (MOOC) platforms spend considerable time searching for courses and learning resources. MOOC platforms contain hundreds or thousands of courses available to learners worldwide. In 2020 and 2021, approximately 60 and 40M learners signed up for at least one MOOC [3]. With the increasing number of learners, the numbers of courses and instructors also increase. Then, finding suitable courses on these platforms can become overwhelming.

Another challenge of MOOCs is that the drop-out rate is 90% [4, 5]. Numerous things could cause learners to drop out of courses, but this trend is negative regardless of why they drop out. To assist users in discovering courses that align with their interests, RSs can be used. This could lower the drop-out rates and enhance the overall learning experience.

Even though the use of RSs has emerged to solve the presented problems, they are not flawless. They experience challenges connected to the data the recommendations are based on and the entering of new users, to mention some. By mitigating these challenges, the recommendations can be improved. In addition to helping users find their desired content more efficiently, the RSs should be enhanced to handle the continuous increase in data on the Web.

Most RSs base their recommendations on numerical ratings. As stated by Al-Ghuribi and Mohd Noah [6], recommending items solely based on their overall rating can lead to inaccurate ratings since these may not accurately represent the users' preferences. Further, as only the overall rating is considered, a lot of available data is not used for recommendation. Therefore, the addition of other features has been researched. For example, in Course Recommendation (CR) much information about the MOOCs, learners, and instructors is available. Zheng et al. [7] express that users explain their ratings through reviews. Hence, some of the meaning could be lost if an RS only considers the numerical ratings when recommending items. If the reviews are taken into account, the recommendations can be improved.

Several approaches have been followed to include user reviews in the recommendation process. One of these is to combine actual ratings with ratings inferred from the reviews to enrich the original ratings. This is usually done by extracting the sentiment of the reviews through Sentiment Analysis (SA). Then, the two ratings can be combined using weights [8–10], the average of the two [11], or other approaches [12, 13]. To the best of the author's knowledge, no research combines actual and inferred ratings in CR using weights.

1.2 Research Goals and Questions

This research aims to assess the potential improvement in the recommendation of MOOCs by incorporating SA techniques to adjust the ratings derived from course reviews. To evaluate this incorporation thoroughly, different recommendation algorithms were used. By analyzing the impact of sentiment-based adjustments, the study aims to enhance the performance and ranking accuracy of course recommendations for learners, ultimately improving their learning experience and satisfaction. However, this research was time-limited as it was conducted in a master's thesis. Therefore, the following research questions were identified with this limitation in mind.

RQ1 How does incorporating sentiment analysis of course reviews, and adjusting rating values based on sentiment, affect the performance and ranking accuracy of course recommendations in MOOCs?

RQ2 To what extent does the choice of rating-based approaches impact the performance and ranking accuracy of popularity-based recommender systems for course recommendation in MOOCs?

RQ3 How is a popularity-based recommender system's performance and ranking accuracy with various rating-based approaches affected by incorporating sentiment from course reviews?

The following should be noted about the research questions. First, for the second research question, a rating-based approach means how items are measured as popular. This could be the average or number of ratings for online courses, for example. Additionally, the third research question combines the two first by looking at the impact the adjusted ratings have on the rating-based approaches.

1.3 Contributions

The research in this thesis contributes to the field of CR and the understanding of rating-based approaches in popularity-based RSs. By incorporating sentiment, more information provided by the users could be exploited to generate recommendations. Experiment 1 contributes to the understanding of how the RS's performance is affected by considering textual reviews in the recommendation. An RS incorporating sentiment from course reviews into recommendation is proposed. The research also contributes to the way in which the sentiment is combined with the actual ratings. By adopting methods used in other domains, this research provides a thorough analysis of the impact on the recommendation of MOOCs.

Moreover, Experiment 2 contributes to the understanding of popularity-based RSs through the focus on finding the most popular items in various ways. Several rating-based approaches are proposed and compared. The takeaways from the experiment can be used in other domains than education, as popularity-based RSs are used broadly. Finally, Experiment 3 contributes to the domain of CR and popularity-based RSs as it combines the two former experiments. An analysis of the impact of recommendations in popularity-based RSs when incorporating sentiment is provided.

1.4 Thesis Outline

The following is an overview of the thesis, including descriptions of each chapter.

Chapter 1 - Introduction	The current chapter introduces the thesis, including its motivation, goals, research questions, and contributions.
Chapter 2 - Theoretical Background	Contains the necessary theoretical background to follow the work done in the thesis, focusing on RSs and Natural Language Processing.

Chapter 3 - Related Work	Gives an overview of research related to Course Selection, RSs in education, and review-based and popularity-based RSs.
Chapter 4 - Data	Presents the dataset used for evaluation in this research, as well as preprocessing and splitting of the data.
Chapter 5 - Method	Gives an overview of the system architecture and the approach followed in the thesis.
Chapter 6 - Experiments	Describes the experimental plan and the implementation details of the experiments.
Chapter 7 - Results	Presents the results obtained in the experiments.
Chapter 8 - Discussion	Contains the discussion of the results related to previous research and expectations.
Chapter 9 - Conclusion	Wraps up the research conducted in this thesis, along with some comments on possible future work.

Chapter 2

Theoretical Background

The following chapter contains the necessary theoretical background to follow the discussions and reflections in the thesis. First, some concepts related to Recommender Systems are introduced in Section 2.1, followed by Natural Language Processing in Section 2.2.

2.1 Recommender Systems

RSs have emerged to provide the users with the most relevant content, as introduced in Section 1.1. The aim of RSs is to recommend relevant content to users based on their previous interactions with a system. Aggarwal [14, p. 3] states that the recommendation problem may be formulated differently. On one side, the *prediction* model tries to predict the rating a user would give to an item. Conversely, the *ranking* model recommends the top-k items to a user. In this approach, it is not necessary to predict the exact rating a user would give an item but to rank the best possible options.

This section continues with a description of a set of recommendation techniques in Section 2.1.1. Later, some challenges in RSs are elaborated on in Section 2.1.2, before some thoughts about evaluating RSs are given in Section 2.1.3. Finally, the usage of RSs in specific domains is discussed in Section 2.1.4.

2.1.1 Recommendation Techniques

RSs are used to predict and suggest items that a user may be interested in. There are various techniques used in RSs, each with its own strengths and weaknesses. Here, the techniques used in this thesis are presented.

Ranking

Ranking-based RSs recommend items to users based on a ranking of their relevance to the user's preferences. The RS generates a list of items that are likely to be of interest to the user and then ranks them according to how relevant they are

to the user's preferences. In the end, the items at the top of the list are considered the most relevant and recommended to the user.

Popularity-based recommendation models are a type of ranking-based recommendation model. These models suggest items to users based on popularity or trends. As they do not consider the users' preferences, they are helpful in situations where there is limited user data available or to promote popular items. Therefore, they can, for example, be used when dealing with new users, as there is no data about them. Several aggregation functions are used to decide which items are the most popular. For example, one can look at the number of users that have reviewed or interacted with an item or the item's average rating. Hence, popularity-based models use historical data to determine the most popular items.

Collaborative Filtering

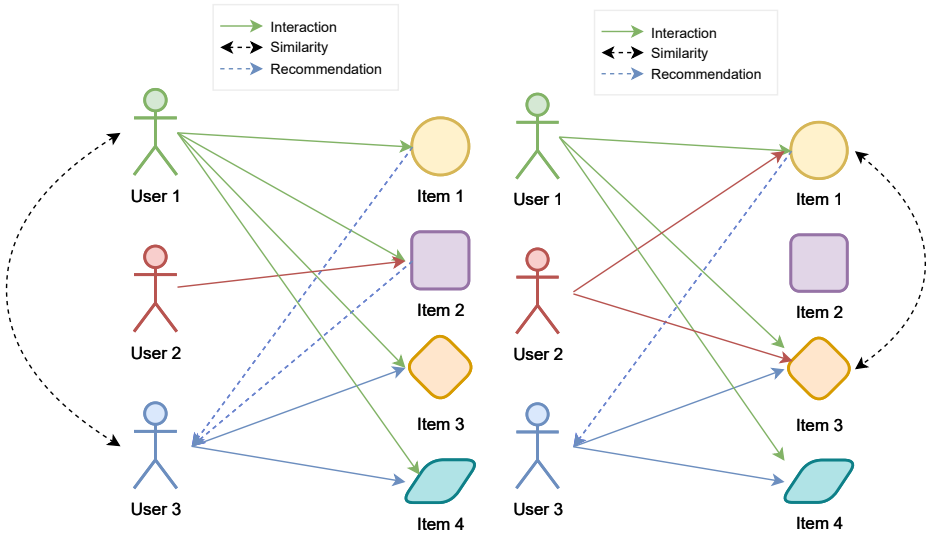
In Collaborative Filtering (CF), items are recommended to users based on similarity. There are two types of CF; user- and item-based. User-based CF recommends items to a user based on similar users' preferences. Imagine that a user usually takes online courses related to web development. Then, the recommender model can find users who have taken similar courses and recommend courses these have taken but that the target user has not. This idea is based on the assumption that users with similar tastes in the past are likely to have similar tastes in the future. On the other hand, item-based CF recommends items similar to items the user has previously liked. More information about CF can be found in [14, p. 8].

Figure 2.1 shows examples of both types. Figure 2.1a displays an example of user-based CF. As Users 1 and 3 are similar, because of their interactions with Items 3 and 4, User 3 is recommended Items 1 and 2. Hence, the recommendations are based on what a similar user prefers. However, in Figure 2.1b, an example of item-based CF is shown. Here, User 3 is recommended Item 1 because of the similarity between Item 1 and Item 3. The similarity of the items is based on the fact that both User 1 and User 2 like them.

The user-based approach tends to be more accurate when there are fewer items than users, while the opposite applies to the item-based approach. However, both approaches can suffer from sparsity and cold start problems. These CF-specific problems are discussed further by Su and Khoshgoftaar [15].

K-Nearest Neighbors

The k-Nearest Neighbors (k-NN) algorithm is a classifier that can be used in RSs. It classifies or predicts data points based on their proximity to neighboring points. This is done by grouping together k of these points in clusters. The algorithm was first introduced by Fix and Hodges [16] before Cover and Hart [17] expanded on the theory. These papers go through the algorithm in detail. In recommendation, k-NN is used as a type of CF algorithm as it classifies the items based on users' previous interactions. Then, the preferences of the users in the same groups are used to generate recommendations.



(a) User-based CF. **(b)** Item-based CF.

Figure 2.1: User and item-based Collaborative Filtering. Solid lines represent interactions, dotted lines with two arrows represent similarity, and dotted with one arrow represent a recommendation.

Matrix Factorization

Matrix Factorization (MF) is another technique used in CF for RSs. The idea is to represent the user-item matrix in two lower-rank matrices, one for the user preferences and the other for the item features. Then, these two matrices are multiplied together to reconstruct the original user-item matrix. The resulting matrix has values that can be used to predict the likelihood of a user interacting with an item. Thus, the technique allows the RS to infer user preferences through *implicit feedback*, described in Section 2.1.3. Therefore, MF handles large and sparse datasets well. Funk originally presented the idea in his blog post [18] during the Netflix Prize challenge [19]. Later, it was further developed and can be read about in various papers, such as by Koren et al. [20].

Alternating Least Square (ALS) is a type of MF that uses an alternating optimization strategy to factorize the matrix and learn latent factors for users and items [14, p. 105]. It is most commonly started with random initial values to avoid being stuck in a local optimum and to explore a wider range of solutions [21].

Content-Based Filtering

Another algorithm often used in RSs is Content-Based Filtering (CBF). It recommends items to users based on the characteristics of the item. Each item is represented by a set of characteristics, such as a course's title, description, instructor, average rating, etc. The recommender model bases its recommendations on the

similarity between a user's preferences and an item's characteristics. Figure 2.2 shows an example of CBF. The two items are similar, represented by the bidirectional arrow. As User 1 likes Item 1, they are recommended Item 2 because of the items' similarity.

Compared to CF, CBF has other advantages. Firstly, CBF can recommend items that are not popular among other users but match better with a user's preferences. Secondly, CBF is less vulnerable to the cold start problem when new users enter the system, as it rather looks at the similarity between a user's preferences and an item than the similarity between users. A third advantage over CF is the ability to explain the recommended items based on their attributes. However, CBF can suffer from the overspecialization problem, which leads to a lack of diversity in recommendations. For more information about CBF, see [14, p. 14].

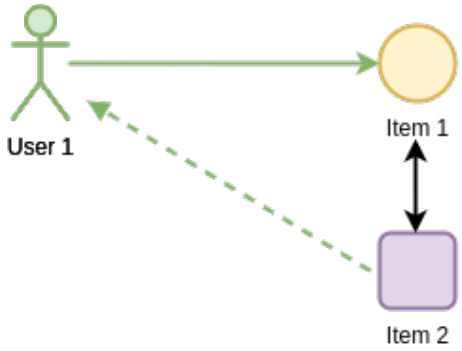


Figure 2.2: Content-Based Filtering.

Hybrid

Hybrid RSs are combinations of different recommendation techniques. They were introduced to overcome the shortcomings of individual techniques. By combining them, the strengths of each can be exploited, and some of the weaknesses omitted. How the different algorithms are combined depends on the system's creators and is affected by the purpose of the system, among others. As an example, a CF technique could be combined with a ranking-based RS through top-N recommendations to get a ranked list of recommendations. Hybrid approaches are further explained in [14, p. 19] [22].

2.1.2 Challenges in Recommender Systems

RSs face several challenges that can make it difficult to provide accurate and useful recommendations to users. By overcoming these challenges, RSs can provide users with more accurate and useful recommendations, increasing engagement and satisfaction. Here, the challenges relevant to the thesis are described.

Data Sparsity

One of the main challenges is *data sparsity*. This occurs when insufficient data is available to predict user preferences accurately [15]. This can be particularly problematic for new items with limited user data available. In online courses, the sparsity problem can occur when many courses are available, but each course only has a small number of enrolled students. As mentioned in Section 2.1.1, CF is vulnerable to sparsity problems.

Overfitting

Overfitting is another challenge for RSs, where the model becomes too specialized to the training data and performs poorly on new data [23]. This can lead to poor generalization and inaccurate recommendations. Two factors that can be adjusted to avoid overfitting are the training data size and the training process duration. For example, suppose the RS on an online course platform relies too heavily on past user behavior to make recommendations. In that case, it may not adapt well to users that change their preferences.

Scalability

Scalability is another challenge for RSs, especially as the number of users and items in the system grows. The computational complexity of the algorithms used to generate recommendations can become a bottleneck, making it difficult to scale the system to handle large volumes of data. This could happen in any platform, for example, online course platforms, because of the growing number of users or courses. More information about scalability can be found in [15].

Cold Start Problem

Another challenge is the *cold start problem*, which occurs when a new user or item is added to the system and there is not enough data available to make accurate recommendations [14, p. 24] [15]. This can be particularly challenging for CF methods, described in Section 2.1.1, as they rely on user behavior data to make recommendations. Imagine that a new user signs up for an online course platform. Then, the RS may not have enough data to generate recommendations for this user accurately. Or that a user has not taken any courses yet, so the RS does not know enough about the user to recommend courses.

Overspecialization

Overspecialization refers to the problem where an RS only recommends items that are similar to the ones the user has already liked [22]. This leads to a lack of diversity in the recommended items, as the users are not exposed to new content. Overspecialization can happen when a CBF algorithm is used, as it tends to recommend items based on the similarity between their characteristics and the

user's preferences. An example of overspecialization is if a user who has taken web development courses only gets recommended other courses about this topic. The user is likely interested in other topics, so recommending diverse courses is important to increase user satisfaction and engagement.

2.1.3 Evaluation of Recommender Systems

Evaluating an RS is essential to measure its effectiveness in providing relevant and accurate recommendations. This evaluation is necessary to ensure that the system performs as intended and meets the needs of its users. In addition, evaluation can identify areas where the RS can be improved, for example, by identifying patterns in user feedback that indicate dissatisfaction with the system. Following, different types of evaluation and feedback are explained. In addition, some evaluation metrics are presented.

Online and Offline Evaluation

Evaluation of an RS can be done either with *online* or *offline* methods [14, p. 225]. In online methods, the users directly participate based on the generated recommendations. In offline methods, on the other hand, the users participate based on a previously obtained dataset. An advantage of the former is the possibility of observing the users and asking why they act as they do. Further, online evaluation requires more time and resources, as the users are followed up manually, whereas offline evaluation can be performed whenever it suits the participants. Also, in offline evaluation, the data the RS is tested on can easily be changed to ensure a generalized system. Whether online or offline evaluation suits the best depends on the task and resources at hand, although offline evaluation is by far the most common from a research perspective [14, p. 226].

Explicit and Implicit Feedback

Users provide feedback continuously when interacting with a system. *Explicit* feedback is provided directly from the users, for example, through ratings or comments [14, p. 347]. On the contrary, *implicit* feedback is information that can be interpreted through user actions, such as clicking on a link. These are actions where one cannot know for sure that the user, for example, watched the whole video or read the whole article. In addition, whether the user enjoyed the content is unknown. Several examples of explicit and implicit feedback are given in Table 2.1.

Explicit feedback is more descriptive and meaningful than implicit feedback. However, it is more time-consuming to obtain the former, as the user needs to spend time giving feedback. Implicit feedback, on the other side, is more accessible but needs more follow-up work and interpretation to be usable. Therefore, the decision of which type of feedback to use in evaluation depends on the goal and resources of the project.

Table 2.1: Examples of explicit and implicit feedback.

Type of feedback	**Action**
Explicit	Write textual comment
	Give rating
	"Like"/"Dislike" an item
	Give feedback in questionnaire
Implicit	Clicking on a link
	Watching a video
	Reading an article
	Time spent on a website

Evaluation Metrics

Evaluation metrics play a crucial role in assessing the performance of RSs as they evaluate the quality of the provided recommendations [14, p. 226]. Several types of metrics, for example, decision support and ranking metrics, measure different aspects of the system. It is essential to use multiple metrics to provide a comprehensive evaluation of an RS's performance.

Decision support metrics measure how well an RS helps users make good decisions. *Precision* and *recall* are examples of these. The former measures the percentage of recommended items relevant to the user [24]. Its formula is presented in Equation (2.1), where r_k is the number of retrieved relevant documents at k and k is the number of specified documents. Recall, on the other hand, measures the percentage of relevant items the system recommends. Its formula is present in Equation (2.2), where r_k is the number of retrieved relevant documents at k and R is the total number of relevant documents.

$$\text{Precision@k} = \frac{r_k}{k} \tag{2.1}$$

$$\text{Recall@k} = \frac{r_k}{R} \tag{2.2}$$

Both of these metrics are represented with values between 0 and 1; the closer to 1, the better. Additionally, both can be specified to look at the k first recommendations. As an example, *precision@10*, where k is 10, measures the precision among the ten first recommended items.

A shortcoming of the decision support metrics is that they fail to capture the order in which items are recommended. The aim of *ranking* metrics is to measure an RS's ordering of items based on how it would have been ordered by the user [14, p. 242] [25]. Normalized Discounted Cumulative Gain (nDCG) is a ranking metric proposed by Järvelin and Kekäläinen [26] that values putting highly relevant documents high up the recommended lists. Its primary advantage is that

it considers the graded relevance values. This is useful for datasets where the relevancy of the items is not boolean but rather on a scale. The equation for nDCG is provided in Equation (2.3), where DCG_k is Discounted Cumulative Gain (DCG) at k and IDCG_k is the Ideal Discounted Cumulative Gain at k (max possible DCG at k). DCG weighs each relevance score based on its position. Hence, the items at the top of the list get a higher weight. A shortcoming is that the relevance score depends on the number of items, meaning a recommender with more items is more likely to have a higher score. An expansion of this is the Normalized Discounted Cumulative Gain (nDCG) metric. It normalizes the values to mitigate the downside of DCG.

$$\text{nDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k} \tag{2.3}$$

Another ranking metric is Mean Average Precision (MAP). The Average Precision (AP) is the average of the precision scores computed after each relevant document is retrieved. In this way, it considers the order of the retrieved items. Thus, MAP looks at two aspects: the relevancy and order of the recommended items. However, it does not consider how relevant the recommended items are. This downside is mitigated if the number of recommendations is relatively low. Its formula is provided in Equation (2.4), where r is the position of a relevant document and R is the total number of relevant documents. Then, to get the MAP, the AP is averaged over many queries. More information about MAP can be found in [14, p. 246].

$$\text{AveragePrecision} = \frac{\sum_r \text{Precision}@r}{R} \tag{2.4}$$

2.1.4 Recommender Systems in Different Domains

RSs play an important role in our everyday lives. They are used in various applications, such as movies, music, news, and social media. Here, the usage of RSs in the educational domain is described and compared to other domains.

Recommender Systems in Education

In the educational domain, RSs are used to recommend items such as learning resources and courses. The recommendation of learning resources can be focused on teachers to find the best resources for their students or on students to help them find the best resources. In Course Recommendation (CR), the aim is to help learners find the most suitable courses and for them to spend as little time as possible doing so.

CR is used in several types of education, as discussed by Urdaneta-Ponte et al. [2]. Some RSs focus on *formal* education through recommending university courses. In contrast, others focus on *non-formal* education, for example, by recommending courses to users on MOOC platforms. Non-formal education differs

from formal education because it evolves around learning individuals do outside the school system. For further explanations and examples about types of education, see [2].

Compared to a university and its courses, a MOOC platform contains hundreds or thousands of courses available to users on the Web. These can be taken by many users simultaneously, which raises challenges such as scalability, explained in Section 2.1.2. With the massive number of courses available, recommending courses to the users is even more critical. Then, users can take courses suitable for them instead of being overwhelmed and spending a lot of time searching for courses.

Comparison with Other Domains

As mentioned, RSs are used in many domains. Two well-known RSs are Netflix's[1] movie recommender [14, p. 5] and Amazon's[2] recommender [27]. According to Netflix [28], using RSs on their platform reduces the time spent searching for movies and series and helps the users find shows and movies of interest. On the other hand, Amazon developed its own recommendation algorithm because the existing algorithms could not scale to their needs [27]. They also need to immediately react to changes in a user's data and provide meaningful recommendations to users regardless of the number of items purchased. The goals of RSs in eCommerce are to increase revenue, build brand trust, and convert visitors into buyers[3].

The aims of the different domains have both similarities and differences. Netflix's aim of reducing time spent searching for content is similar to the aim of CR of helping learners spend as little time as possible searching for courses. Even though some MOOCs are priced, the application of RSs is not to maximize revenue but rather to help users find the most suitable courses. This aim differs from the aim of eCommerce of increasing revenue. Thus, the aim of applying RSs differs for various domains.

2.2 Natural Language Processing

Natural Language Processing (NLP) is a part of Artificial Intelligence (AI) that focuses on making computers able to understand, interpret, and generate human language. Thus, it helps improve communication between humans and machines. NLP involves developing algorithms and models that can process and analyze text similarly to how humans understand language.

Some text preprocessing techniques are described in Section 2.2.1. Then, Language Models and Sentiment Analysis are presented in Sections 2.2.2 and 2.2.3.

[1] https://www.netflix.com/
[2] https://www.amazon.com/
[3] https://influencermarketinghub.com/ecommerce-recommendation-system/

2.2.1 Text Preprocessing

Natural language is *unstructured*, meaning there are no predefined rules for the data format. In addition, it is noisy and can contain typos because of human errors, which makes it difficult for machines to interpret it. Therefore, human language has to be preprocessed for machines to understand it. Text preprocessing aims to clean and transform the raw text into a format more suitable for NLP analysis. Which preprocessing steps to perform depends on the situation and below, some of these steps are described.

Noise Reduction

The human language has many meaningless characters, especially in the "social media" language. One step of textual preprocessing is *noise reduction*. This accounts for removing special characters, punctuation, and numbers irrelevant to machines. Many English words are combined with an apostrophe to create a *contraction*. One way of handling these is to expand the contractions back into the words they consist of. An example is "we'll" which can be expanded into the words "we will". Digits can also be replaced by their textual representation, for example replacing "20" with "twenty". Which characters or words are noisy depends on the task at hand. Other noisy examples are text in languages other than the desired one, abbreviations, and erroneous data.

Normalization

Another preprocessing step is the *normalization* of text. This is the process of converting a token into its base form. This makes the data more consistent and more manageable for machines to process. One aspect of normalization is the casing of letters. As lowercase and uppercase words are assumed to have no difference, all letters should be converted to the same casing [29]. By lowercasing all words, "English" and "english" would be handled as the same, disregarding typos. Further, extra spaces should be removed as they can create extra noise in the NLP process.

Another normalization step is *lemmatization*. It is the process of reducing different forms of a word into its meaningful base form [30]. Compared to *stemming*, which removes the last characters from a word [14, p. 145], lemmatization bases its reduction on the context. A comparison of the two is visualized in Table 2.2. The two words "information" and "informative" are both turned into the word "inform" for the stemmed version, whereas the words are unchanged in the lemmatized version because of the context. More information about normalization can be found in [30].

Stop Word Removal

Stop words are non-informative high-frequency words that can be removed from the text [31]. These include articles, prepositions, pronouns, and conjunctions.

Table 2.2: Stemming and lemmatization of a selection of words.

Word	Stemmed	Lemmatized
information	inform	information
informative	inform	informative
computers	comput	computer
feet	feet	foot

Stop words vary from language to language and task to task, but have in common that they do not carry information. Some English words that usually are considered stop words are "the", "as", and "or". The data size could be reduced by removing these words, and the accuracy of NLP models increased.

2.2.2 Language Models

A Language Model (LM) is an AI system that has learned the patterns and structure of languages. It can generate human-like text based on input and perform tasks such as question-answering, language translation, and creative writing. The topic has been researched extensively in recent years, leading to the introduction of BERT and GPT, which are presented here.

BERT

Bidirectional Encoder Representations from Transformers (BERT) was presented as a language representation model in 2019 by Devlin et al. [32]. It was trained *unsupervised* on a large corpus of English data. In this way, BERT learned a representation of the English language, which enabled it to obtain good results on NLP tasks.

The *B* in BERT stands for *bidirectional*. This means the model reads all words in a sequence simultaneously, meaning it reads from both left and right. In this way, it can understand the context of texts. The *T* stands for *transformers*, a mechanism presented in 2017 by Vaswani et al. [33]. Its goal is to learn the contextual relations between words in a text. Therefore, to make BERT perform as best as possible, removing stop words discussed in Section 2.2.1 may not be needed.

GPT

Another type of LMs is Generative Pre-trained Transformer (GPT) models. These are also based on the transformers architecture, as BERT. However, in contrast to BERT being bidirectional, they are *autoregressive*. This means they generate text by predicting the next word given the preceding context. The models were introduced by OpenAI[4], respectively GPT-1 in 2018 [34], GPT-2 in 2019 [35], GPT-3 in 2020 [36], and GPT-4 in 2023 [37].

[4]https://openai.com/

2.2.3 Sentiment Analysis

Sentiment suggests a settled opinion reflective of one's feelings, according to the Merriam-Webster Dictionary[5]. Sentiment Analysis (SA) evolves around computationally deciding a text's sentiment, opinion, or subjectivity [38]. Often, SA is performed on shorter texts where people express thoughts, such as tweets or reviews. The output from a model that performs SA can vary, from the most common positive vs. negative labeling, to, for example, a rating of 1-5. The process can be used to learn more about the text's authors or intercept information.

SA can be performed using LMs or other methods such as ML algorithms or rule-based methods. The steps of SA include preprocessing, tokenization, and pretraining before the model can predict the labels. Pretraining a model means training it to perform NLP tasks. How this process is done affects the results of the labeling process. If the model has been trained too much on the same type of data, it can become overfitted, as discussed as a challenge for RSs in Section 2.1.2. When the model has predicted the labels, it is evaluated using metrics that, for example, could be the ones presented in Section 2.1.3.

[5]https://www.merriam-webster.com/

Chapter 3

Related Work

The current chapter gives an overview of related research done in the domain. First, research related to Course Selection is presented in Section 3.1. Then, the use of RSs in the educational domain is discussed in Section 3.2. Further, research on review-based RSs is presented in Section 3.3 and rating-based approaches in popularity-based RSs in Section 3.4.

3.1 Course Selection

Course Selection (CS) revolves around selecting courses in different environments, such as in universities or on MOOC platforms. Finding the courses to follow is tedious, and many factors affect the process. On online platforms, the courses are available for everyone. However, some may be advanced courses expecting the users to have specific prerequisites. Several studies have been conducted on the topic of CS. Most of them are focused on formal education but are still applicable to CS on MOOC platforms.

Based on the work of Kerin et al. [39], the students' significant considerations during CS are their interest in the area, the course's content, the instructor, and the course's compatibility with the field. Even though this study was conducted almost 50 years ago, the results are still relevant for formal and non-formal education and CS.

Several researchers discuss how grades affect the selection of courses. Ognajovic et al. [40] conclude that the most crucial factor is how a student's Grade Point Average (GPA) is compared to the course average. Another factor they discuss is how a course correlates with the student's career objectives. Other studies that mention grades as an essential factor are [41, 42]. Though, the focus on GPA is not relatable to CS of MOOCs. MOOCs could be used as support for learning the contents of courses in formal education, but as they are not graded, they are not chosen based on this.

Lynn and Emanuel [42] focus on eight factors that impact CS. These are personal interest, simplicity, future career goal, course format, instructor, social

status, schedule, and examination protocol. The first five could also be related to CS of MOOCs, whereas the last three are linked explicitly to formal education. The study does not conclude which is the most crucial factor, as this depends on the individual.

Babad and Tayeb [43] divide CS into two groups related to academic and personal considerations. The former group is focused on course and course characteristics. These involve the instructor's mode of teaching, personality, style, and the course's academic quality. The factors related to the instructors are also highly relevant for MOOCs.

According to Zheng et al. [44], 65% of courses in the fourteen platforms surveyed grant course certificates upon completion. As certifications could be used as proof of knowledge one has obtained, it is a motivating factor in CS of MOOCs. As stated in the study, some universities in China also recognize MOOC certificates with credits. This way, completing MOOCs could also be a motivating factor during the completion of an educational degree.

Some considerations should be taken based on the literature in the domain of CS. First, there are many factors affecting the selection of courses. Some relate to the learner, such as personal interests, prerequisites, and career objectives. On the other hand, some are related to the course and instructor, for example, the instructor's teaching style and personality. As which factors are the most crucial depends on the individual, the target users' motivations should be kept in mind while developing an RS.

3.2 Recommender Systems in Education

RSs have found applications in various domains such as movies, online shopping, traveling, news, social media, and more, as introduced in Section 1.1. In the past decade, the use of RSs has also extended to education. Following, some of the various research topics RSs are used for in education are presented.

One research topic is the recommendation of study sequences and learning paths. Pang et al. [4] recommend subsequent courses based on the learner's history. Hou et al. [45] cluster courses based on prerequisite dependencies. Then, they recommend courses after the ones the learner has taken. Another approach is followed by Zheng et al. [46]. Based on their descriptions, they construct a Knowledge Graph (KG) of courses. Later, they find a learning path for the learner based on their interests.

Other studies focus on helping students and teachers with learning and teaching. Some studies look into the recommendation of learning resources, such as [47, 48]. Hajri et al. [49] help learners learn a course's prerequisites by recommending open educational resources. They also provide resources to ensure the learner has understood the material after completing a course section. Another research topic is the recommendation of resources for teaching practice. For example, Sergis and Sampson [50] aid teachers in their selection of learning objects.

The largest proportion of research on RSs in education is related to Course Recommendation (CR). CR is present in formal and non-formal education, introduced in Section 2.1.4. Below, research in the two domains is presented.

3.2.1 Course Recommendation: Higher Education

Students commonly have to select some elective courses in formal education, such as universities or other higher educational institutions. The information available about these courses differs significantly in quality and amount for different institutions. To make the CS process as easy as possible, RSs are used. Many approaches have been tested to simplify the CS process; some are based on the factors discussed in Section 3.1. Research related to CR in higher education is presented here.

A commonly used approach in CR is the prediction of grades. Olapido et al. [41] uses logistic regression to predict the score they will obtain by taking the course and use this to predict if the student will take the course. Ceyhan et al. [51], on the other hand, use a CF approach to predict grades. They use information about previous students and their obtained grades to predict grades for students based on similarity. The process of recommending courses based on grades works well for students that care a lot about their GPA. However, it may have flaws for students who select courses based on other factors, such as personal interest, schedule, or instructor.

Instead of using CF to find users with similar grades, the technique could be used to find users with other similarities. For example, Bhumichitr et al. [52] looked at student similarities through their course templates. They created user profiles based on students' academic records. Then, they recommended the courses similar users had taken. This approach is more focused on students' interests and personalities. By recommending courses based on shared interests, the students will likely get more intriguing recommendations but have to do some practical work themselves by looking up the schedules and course formats, etc. Deraman et al. [53] also focused on the students' personalities and interests. However, the implementation is poorly developed through if-then-statements and only tested on twelve students.

Another approach to recommending courses is through social constraints. Channarukul et al. [54] use information from Facebook[1] in the recommendation. They recommend course schedules based on the schedules of the students' Facebook friends. This approach is attractive as much information is available on the Web. Although, because of GDPR, such information must be provided with consent from all involved parties.

Lastly, hybrid approaches have been used to develop RSs in CR. Unelsrød [55] created an RS that recommended courses to students at NTNU based on data from the course evaluation website *Classmate*. It combined CF and CBF techniques to overcome the sparsity problem explained in Section 2.1.2.

[1] https://www.facebook.com/

3.2.2 Course Recommendation: MOOCs

The amount of courses on MOOC platforms has increased massively since "The Year of the MOOC" [56] in 2012. At the same time, the drop-out rate on these platforms is 90% [4, 5]. Further, Gomez et al. [57] observed that the decision-making process is challenging for learners as existing MOOC platforms contain courses with extremely positive ratings. With these challenges in mind, some research focused on CR of MOOCs is presented.

Some research on learners on MOOC platforms have been conducted. Gütl et al. [5] investigated the reasons why learners drop out of MOOCs. They asked the subjects about personal and academic reasons, among others. Regarding academic reasons, 70% emphasized that it was challenging to study and work simultaneously, 14.93% that they were not technically prepared for the course, 8.96% indicated that the course was too difficult, etc. Based on the results of this study, there is no definitive answer to why learners drop out of courses. Instead, feedback for courses could be used to improve the courses to lower drop-out rates.

An approach to lower the drop-out rate for MOOCs is to focus on the course prerequisites. Pang et al. [4] state that learners drop out of courses because of prerequisite inadequacy. Therefore, they recommend courses with objectives that cover the prerequisites of a learner's desired courses. Zhao et al. [58] also focus on course prerequisites. They use a neural attention network to combine prerequisites of various courses with the learners' background knowledge. These approaches both help to minimize the gap between learners' knowledge and the prerequisites of other courses.

Other approaches cluster learners based on their interests, personalities, and preferences. Jain and Anika [59] cluster users based on their learning style. *Active* learners are recommended courses based on supervised learning, while unsupervised learning is used for *passive* learners. A potential flaw of this research is the rigid system of two categories. The learners between these two categories will most likely be put in one of the categories pretty randomly. Yanhui et al. [60] clustered users based on preferences and former courses. Then, the information about all learners in a cluster was used to recommend courses to the whole group. This approach exploits information from similar users, meaning there can be some errors in case of outliers. Otherwise, it could be a reasonable mitigation against the sparsity problem explained in Section 2.1.2.

Another focus is the removal of noise and confusing data. Zhang et al. [61] remove courses not contributing when predicting suitable courses for users. For example, unfinished courses or courses taken over a very long time. This approach could be followed as a pre-recommendation step to clean the data. However, as less than 5% of learners complete their courses [62], this approach could make the data even sparser. Thus, this must be taken into account when considering removing data.

Using a Knowledge Graph (KG) has increased in CR in recent years. Some studies combine information from KGs with CF to mitigate its challenges with

sparsity and the cold start problem as elaborated on in Section 2.1.1. Jung et al. [63], for example, complement information from MOOCs with data from an external KG to handle these problems. Zhao et al. [64] focus on dealing with the sparsity problem. They enrich sparse matrices through information from two KGs. The first contains course data and models the courses based on prerequisites. The other one represents a structure of prior and subsequent courses. By combining these, the study aims to improve CR for MOOCs using KGs.

Another focus is dealing with the scalability problem explained in Section 2.1.2. Hou et al. [65] propose an extensive data-supported course RS to deal with the increasing time and space complexities because of the increase in courses. An alternative approach is to narrow the items used to generate recommendations. Garg and Tiwari [66] use CF but focus on a minor part of the users when finding similar users. This decreases the complexity as the generation task is time-consuming when looking at all users.

To sum up, there has been much research on CR. Some focus on mitigating the challenges of RSs, introduced in Section 2.1.2, while others use other approaches to improve the recommendations. Additionally, a wide range of techniques have been used in these applications.

3.3 Review-Based Recommender Systems

Recommending items solely based on the overall rating value could lead to inaccurate ratings, as these ratings may not accurately represent the users' preferences, according to Al-Ghuribi and Mohd Noah [6]. Further, they state that looking at the implicit user preferences through reviews is claimed to be a more accurate approach to determining the users' preferences, as users will write their opinions exclusively regarding the items. Additionally, Zheng et al. [7] express that users explain their reasoning behind ratings through the reviews. Therefore, research combining RSs and reviews is examined.

There are several advantages to including user reviews in the recommendation process. First, the data sparsity problem explained in Section 2.1.2 could be reduced in the case of missing ratings [6]. Further, including reviews can mitigate the cold start problem, also explained in Section 2.1.2. For example, as seen in the work of Wang et al. [67], through enriching users' preferences by predicting the missing information. Reviews can also provide rich information in domains where users' preferences are not well-represented by numerical ratings. Other advantages of using reviews in RSs can be found in [6].

Sentiment Analysis of Reviews in Recommender Systems

Most of the research conducted on Review-Based RSs uses Sentiment Analysis (SA) to extract information from the reviews [6]. Leung et al. [68] inferred ratings through performing SA on movie reviews. This was the first attempt at using information extracted from textual reviews to generate recommendations. The

proposed system was, however, not evaluated. Different approaches to incorporating information from reviews in RSs have been researched. These are presented below.

Infer Ratings From Reviews

Given that no explicit ratings are available, implicit ratings can be inferred from the reviews. Several approaches have been followed to predict missing ratings. Ma et al. [69] utilized information in tweets to predict ratings using MF, while Zheng et al. [7] used neural networks to predict missing ratings. Other researchers have focused on extracting the sentiment related to specific aspects for prediction. Musat et al. [70] established a correlation between review texts and ratings by comparing the predicted opinions of particular aspects to the actual ratings. Further, Yang et al. [71] predicted the ratings using tensor factorization based on the opinions expressed towards different aspects. D'Addio et al. [72] followed this approach using an CF technique for prediction.

Combine Actual and Inferred Ratings

According to Chen et al. [73], combining actual and inferred ratings would likely return better recommendations. Thus, when explicit ratings are available, the implicit ratings could be used to enhance the original ratings. The approach of combining these ratings differs between research. Also, some research adapts based on whether explicit ratings are present or not, while others only deal with the former case.

The most common approach to combining actual and predicted ratings is by summing the ratings using weights. The following works are adaptable regarding the existence of explicit ratings. Osman [8] presents an electronic product RS that predicts ratings using SA on reviews. They are predicted if an item's review does not contain a rating. If it includes an actual rating, the predicted rating is added using the weights 0.3, 0.5, and 0.7. In the end, the weight of 0.7 performs best, meaning the actual rating contributes to 70%, and the predicted to 30% of the summed rating. Zhang et al. [11] also use the predicted ratings explicitly when no actual ratings are available. On the other hand, instead of using weights, they are averaged with the original ratings if present.

The following works combine actual and predicted ratings using weights but only work when explicit ratings are present. Li et al. [9] first denoise the ratings and normalize the sentiment scores from reviews before weights combine the ratings and sentiment scores. Wang et al. [10] also combine the original and predicted ratings using weights. However, they first generate recommendations using a hybrid system with CF and CBF before the score from this is combined with the predicted score from SA of the reviews.

Several other approaches than weights have been conducted to combine actual and inferred ratings. Lee et al. [12] performed SA using the BERT model to predict the sentiment and emotion. Then, they combined the actual and inferred ratings

into a subgraph using a link prediction method before using them in a Graph-Based Movie RS. However, Ling et al. [13] combined the ratings and review texts using LDAMF, proposed by McAuley and Leskovec [74].

Using Sentiment in Course Recommendation

The research presented earlier in this section was evaluated in various domains, such as movies, social media, and e-commerce. However, none of them are related to education. Some research using sentiment in CR is presented here.

Ng and Linn [75] created a CR that bases its recommendations on various features. Among these were a rating predicted by MF and SA of reviews. A back-propagation model combined the features and fed them into the RS. Hazar et al. [76] also predicted ratings using SA of student reviews. Then, they analyzed the differences between the actual and predicted ratings to detect user requirements and interests. The original rating was used during the recommendation for users who have not given a review. Other studies that use sentiment in CR are [77–79]. However, the latter has no implementation, only a proposal of using sentiment scores to boost recommendations.

As learned from the presented research that combines actual and inferred ratings, the most common approach is to combine these using weights. To the best of the author's knowledge, no research combines original and predicted ratings in CR using weights. Thus, this approach can be followed to close this research gap.

3.4 Rating-Based Approaches in Popularity-Based Recommendation

Popularity-Based RSs recommend items based on popularity or trends, as introduced in Section 2.1.1. They tend to be used as non-personalized baselines, as in the research of Jannach et al. [80]. Both because they are computationally inexpensive and since they are hard to beat [81]. However, the approach used to determine which items are popular affects the recommendations significantly. In this section, studies using different approaches are presented.

The most common way of defining popularity in recommendation is by an item's number of ratings [82]. Additionally, Jannach et al. [80], and Rakshit et al. [83] propose that application-specific approaches, such as looking at the number of purchases or accumulated revenue could be used. Numerous studies use this approach, some of them included here [80–86].

According to Cremonesi et al. [81], popularity-based RSs only provide trivial recommendations and are not helpful for either users or content providers. Further, Cañamares and Castells [84] mention that an obvious disadvantage of popularity-based recommendation is the lack of novelty. However, the latter developed a formal analysis of the effectiveness of popularity-based RSs to determine if popularity is effective in the recommendation process. They defined popularity

using three approaches: based on the largest number of ratings, the largest number of positive ratings, and average ratings. The difference between the two first is that the first can give misleading signals as it only looks at the number of ratings but not what the ratings are. Thus, as the second approach looks explicitly at the positive ratings, the aim is to represent better courses that users liked. The same authors also used these two approaches in another earlier research, where they discuss biases in popularity-based recommendation [82]. Further, they propose employing the average rating as a potentially more desirable non-personalized indicator compared to the number of favorable preferences.

Another research that determines popularity using the number of ratings and the average rating is Jannach et al. [80]. They base their choice of rating-based approaches on an analysis of the popularity of the recommended items. They state that the average rating approach is limited by not considering the number of ratings. Looking at the average rating, one cannot know how many ratings the item has. Thus, an item that has gotten one rating of 5.0 would be defined as popular, even though only one person has rated the item.

However, Cremonesi et al. [81] observed that using the average rating resulted in worse accuracy than using the number of ratings. Another strategy is used by Rashid et al. [85]. They balance popularity-based and entropy-based techniques. Popularity-based techniques represent that many users have rated an item, but they do not represent the value of the ratings. On the contrary, entropy-based techniques represent the value of each rating but not the number of ratings received. Thus, they combine the two approaches to get the best of both worlds. To the author's knowledge, this approach has not been used to combine the average rating with other rating-based approaches.

Bellogín et al. [86] prompt some challenges with popularity-based RSs. Their observations are based on the fact that the number of ratings for each item determines popularity. They observe that the recommender would perform at the same level as a random recommender if all items were equally popular. On the other hand, if many users like a few items, and few like the rest, the popularity-based approach will obtain the maximum precision possible. These challenges should be considered when using popularity-based RSs.

Chapter 4

Data

Evaluating the performance of an RS is essential, as described in Section 2.1.3. To evaluate the RS in this thesis, offline evaluation is performed. The dataset to use in the evaluation should be in the domain of online courses and contain information and ratings of the courses and other user interactions, such as textual feedback, that can be used to adjust the ratings. In this chapter, some existing datasets in the educational domain are presented in Section 4.1, followed by Section 4.2 that focuses on the COCO dataset.

4.1 Existing Datasets

There are many publicly available datasets in the domain of online education. Some datasets are related to learning resources, such as the EdNet[1] dataset. As it contains student interactions [87], it is irrelevant to online course selection. Another dataset is the Open University Learning Analytics Dataset (OULAD)[2] [88]. It contains data about courses and students and the relations between them. However, as this dataset only contains data about seven courses and focuses more on students' enrolments, it was ruled out as insufficient.

Kaggle[3], a website for Machine Learning projects and open datasets, has several free datasets on online courses. The EdX Courses Dataset[4] contains course titles, descriptions, and difficulty levels of 717 courses on EdX[5]. As this does not include any course reviews, it was deemed irrelevant. Further, the Coursera Courses Dataset[6] contains course titles, descriptions, difficulty levels, and ratings for 3416 courses on Coursera[7]. Even though this dataset contains ratings, it does not have any other interactions with the user, making it impossible to adjust the ratings.

[1] https://github.com/riiid/ednet
[2] https://analyse.kmi.open.ac.uk/open_dataset
[3] https://www.kaggle.com/
[4] https://www.kaggle.com/datasets/khusheekapoor/edx-courses-dataset-2021
[5] https://www.edx.org/
[6] https://www.kaggle.com/datasets/khusheekapoor/coursera-courses-dataset-2021
[7] https://www.coursera.org/

Hence, it was set aside. The Coursera Free Courses Dataset[8] is another dataset with courses from Coursera. In addition to standard course information, it also contains ratings and the number of reviews for each course. Thus, it has the same flaw as the previous dataset from Coursera, as it does not contain user interactions other than ratings. The same applies to the Udacity Courses Dataset[9], a course dataset from the Udacity[10] platform that also contains standard course information and ratings. Hence, there are many available course datasets, but many only have ratings, not reviews or additional comments.

A dataset focused on course reviews is the 100K Coursera's Course Reviews Dataset[11]. It contains around 140k course reviews scraped from Coursera. These reviews include both a labeled rating and a comment. The distribution of the different ratings in the dataset is shown in Figure 4.1. It is pretty unbalanced, with a lot more positive ratings than negative. This could be because users cannot leave reviews before completing the course [89], and if someone completes an entire course, they likely enjoy it. However, this dataset was discarded because of the small number of reviews with ratings of 1, 2, and 3.

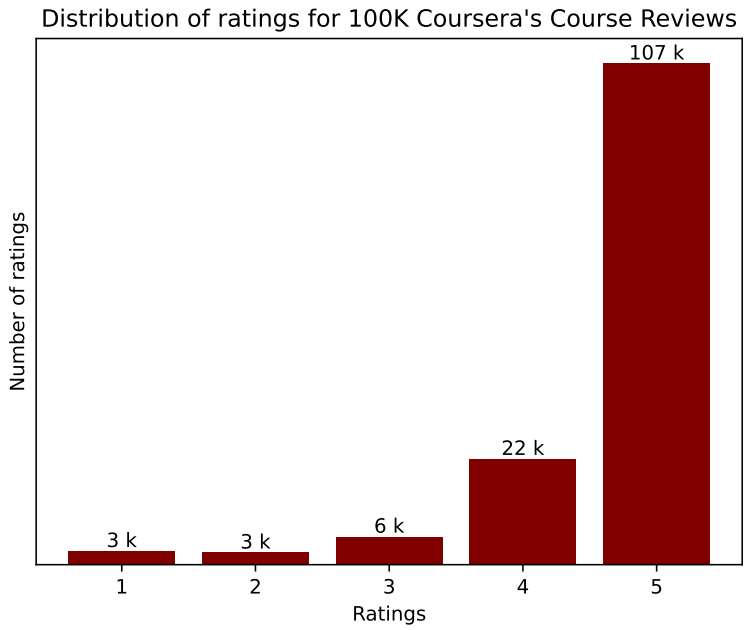


Figure 4.1: 100K Coursera's Course Reviews: Distribution of ratings.

Another dataset in online courses is the Collection of Online Courses (COCO) dataset. It contains data about courses, instructors, and learners from the Udemy[12]

[8]https://www.kaggle.com/datasets/yasirabdaali/coursera-free-courses-dataset
[9]https://www.kaggle.com/datasets/khusheekapoor/udacity-courses-dataset-2021
[10]https://www.udacity.com/
[11]https://www.kaggle.com/datasets/septa97/100k-courseras-course-reviews-dataset?select=reviews_by_course.csv
[12]https://www.udemy.com/

platform. Dessì et al. [90] presented the dataset because of the lack of available datasets in technology-enhanced learning. It contains more than 4.5M reviews for 42k courses. The distribution of the ratings is visible in Figure 4.2. As with the 100K Coursera dataset, it is relatively unbalanced. However, in contrast, it contains many more reviews, so there are many reviews for each rating. In addition, the dataset includes further information about the courses, and some information about the instructors, making it possible to personalize recommendations.

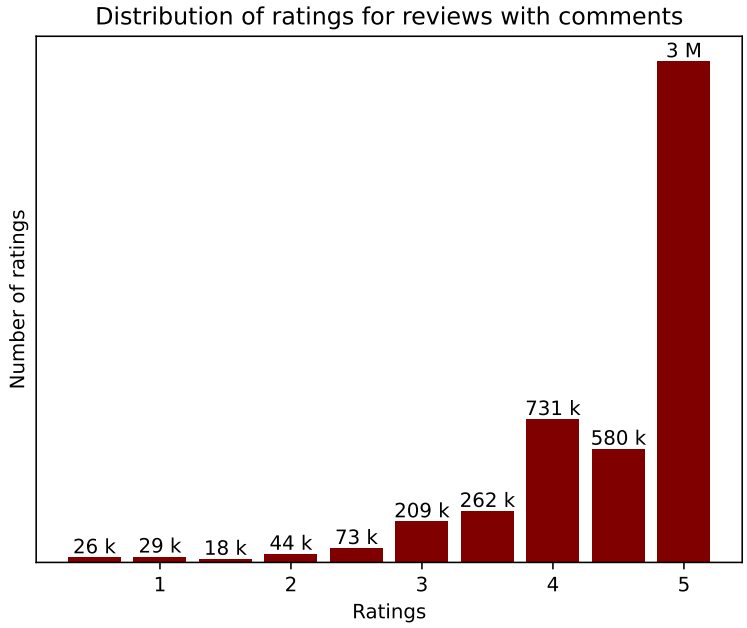


Figure 4.2: COCO: Distribution of ratings.

The examined datasets are compared and summarized in Table 4.1. As can be observed, the COCO dataset contains the most courses and reviews. Because of this, the COCO dataset was chosen to be used in the evaluation.

Table 4.1: Existing datasets in the domain of online courses.

Dataset title	Courses	Reviews	Updated
EdX Courses	717	-	2021
Coursera Courses	3 416	-	2021
Udacity Courses	262	-	2021
Coursera Free Courses	717	-	2023
100K Coursera's Course Reviews	1 835	140 000+	2017
COCO	**42 113**	**4 530 508**	**2017**

4.2 The COCO Dataset

This section describes the dataset in detail, including more characteristics, challenges, preprocessing, and splitting strategy.

4.2.1 Obtaining the Dataset

The dataset is unavailable online but can be obtained by contacting the authors of the COCO paper [90]. Therefore, Dr. Mirko Marras[13] at the University of Cagliari was contacted. Then, a dataset release agreement was signed, containing information on how the data can be used and for which purposes. Ethical issues must be addressed when using data owned by someone else. One cannot use the data in any way not stated in the agreement or discredit the owners of the dataset. Also, the data owners could withdraw the dataset anytime during the research process. This was considered when deciding which dataset to use, but this was considered unlikely. Another essential aspect when dealing with user data is that it should not be possible to identify the users through the data. As this is impossible with the COCO dataset, there was no need to sign a privacy agreement.

4.2.2 Characteristics

The dataset's structure is present in Figure 4.3, with a rotated and enlargened version in Appendix A. After obtaining the dataset, a deep dive was done to get an overview of the data. The insights obtained are summarized in Table 4.2 and further presented in detail in this section.

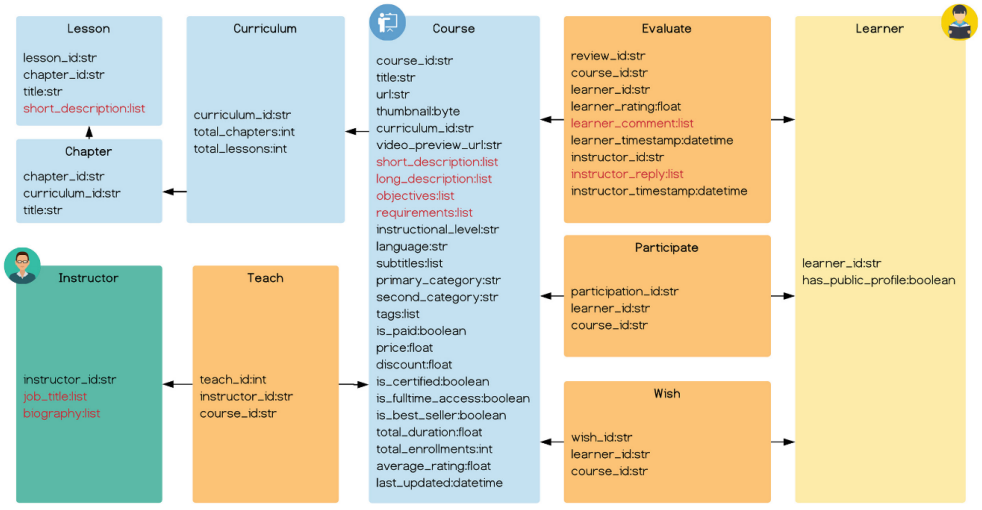


Figure 4.3: COCO: The structure of the dataset, retrieved from Dessí et al. [90].

[13] https://aibd.unica.it/people/mirko-marras

Table 4.2: COCO: Summary of characteristics.

Characteristic	Amount
Courses	42.113
Instructors	16.963
Users	2.426.398
Reviews	4.530.508

Courses

The dataset contains information about more than 42.000 courses on Udemy. The information about each course is shown in the *Course* table in the figure. This includes short and long descriptions, objectives, requirements, and the course's language, subtitles, and target audience. Figure 4.4 displays the 11 languages used in most courses on Udemy. As can be observed, the languages with the most courses are English, Spanish, and Portuguese.

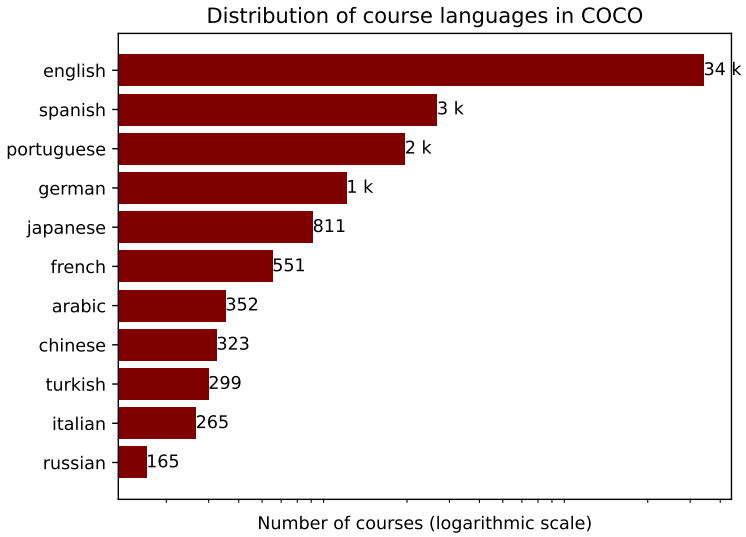


Figure 4.4: COCO: Number of courses taught in each language. The top 11 most popular languages are shown.

The courses are also put into two levels of predefined categories, such as "Business" and "Finance", where the latter is under the former category. Additionally, COCO has specific information about each course's different chapters and sections, such as descriptions of the videos, practices, and quizzes. These are present in the *Curriculum*, *Lesson*, and *Chapter* tables in the figure.

Instructors

COCO also contains data about the instructors of the courses. This is present in the *Instructor* table in the figure. Along with the instructor's ID on Udemy, a description of their job title is present. In addition, the number of courses, enrolments, and reviews the instructor has are also provided. The *Teach* table connects the instructors to the courses they have taught.

Reviews

The most significant proportion of the data in COCO is course reviews, present in the *Evaluate* table in the figure. These are connected to both a course and a user. Additionally, they consist of a rating, possibly a comment, and the timestamp of when the review was given. The ratings are between 0 and 5, and the distribution of the ratings is visible in Figure 4.2. As can be seen, there are a lot more positive ratings than negative ones, but the number of each rating is significant.

Further, Figure 4.5 shows how many reviews contain comments and which do not. Hence, about a third of the reviews have a comment. From these comments, the rating distribution can be seen in Figure 4.6. The distribution looks very similar to the one for all reviews in Figure 4.2, and the number of reviews for each rating is still enough to evaluate.

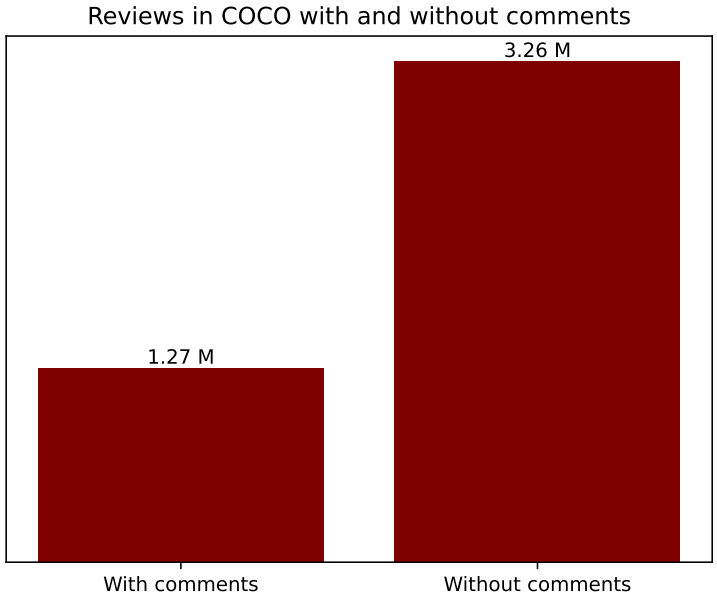


Figure 4.5: COCO: Reviews with and without comments.

As short and low-quality reviews weaken the performance during recommendation [69], the lengths of the comments were looked into. Figure 4.7 shows the distribution of the comment length for reviews in COCO. The intervals span from

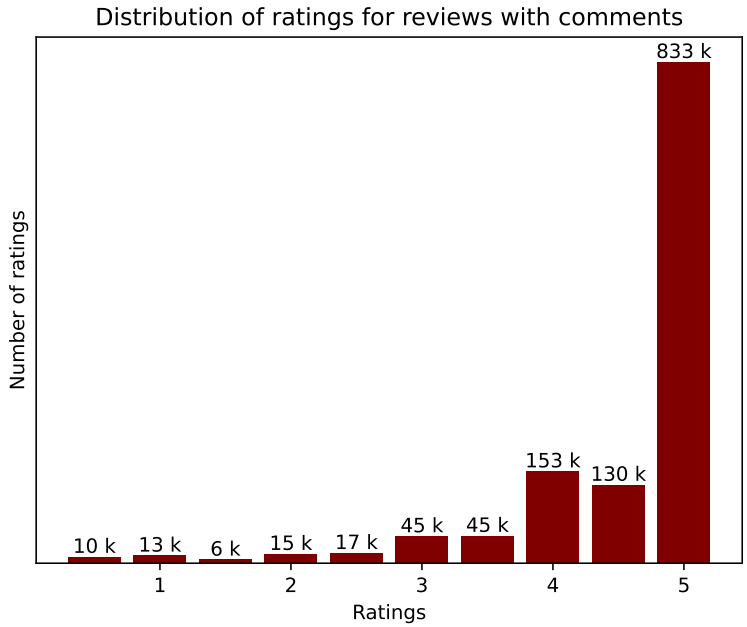


Figure 4.6: COCO Distribution of ratings in reviews with comments.

zero to 250. All comments longer than 250 characters are put into the last bar. As can be observed, the distribution is quite even, and many comments are longer than a few characters.

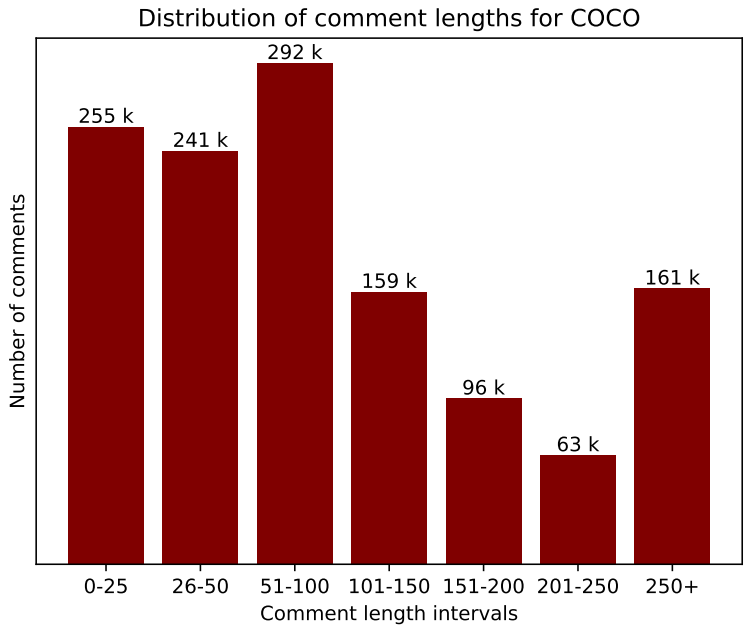


Figure 4.7: COCO: Distribution of review lengths.

Another interesting aspect is how many reviews each user has left. As seen in Figure 4.8, most users have only reviewed one course. Further, when the number of reviews increases, the number of users decreases. However, there are still quite a lot of users that have given more than one review.

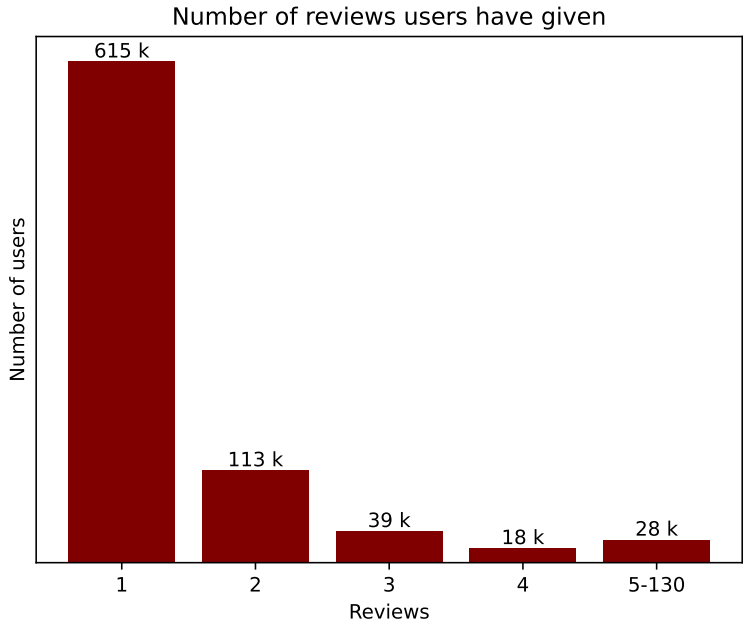


Figure 4.8: COCO: Numbers of users that have given between 1 and 130 reviews.

4.2.3 Challenges and Limitations

As mentioned in Section 2.2.1, there are several challenges when working with human language. Below, some challenges in the COCO dataset are discussed.

Language

As the Udemy platform is available to everyone on the Web, the users and instructors can be from anywhere in the world. As seen in Figure 4.4, most courses are taught in English. Although, there are also many courses taught in Spanish, Portuguese, and German. The courses are taught in different languages, but the reviews are also expected to be in other languages. This challenge should be addressed when working with textual data, as different languages must be handled differently.

Quality of the Data

As instructors and users manually enter the course information and reviews on Udemy, they are prone to errors. For some courses, the data in each attribute are

wrongfully shifted to the following attribute. For example, the short description is entered as the long description; the long description is entered as the course's objectives; the objective is entered as the prerequisites, and so on. In addition, typos and other human errors are present in the data. These challenges should be mitigated through preprocessing.

Distribution of Ratings in Reviews

As addressed earlier in the section, the distribution of the ratings in the course reviews (Figure 4.2) is pretty unbalanced. Schoenmueller et al. [91] analyzed millions of reviews and found that most online reviews are on the positive end of the scale. Hence, this distribution is expected but still needs to be dealt with. If not, it could cause problems when performing NLP tasks such as SA, as the model could perform poorly on negative reviews because of a lack of training data.

Sparsity

As described in Section 2.1.2, data sparsity occurs when insufficient data is available to predict user preferences accurately. The COCO dataset contains around 4.5M reviews, given by more than 800k users. However, as seen in Figure 4.8, 615k users have only reviewed one course. Hence, the data of these users are very sparse, as they only have interacted with one item each.

4.2.4 Preprocessing

Preprocessing helps machines to understand human language and mitigates the challenges presented in Section 4.2.3. When the dataset was obtained, there was no previous data preprocessing. Therefore, the steps on the left-hand side in Figure 4.9 were conducted. These are explained in further detail in Section 2.2.1. The right-hand side of the figure shows an example of a review comment being preprocessed. This example is fictional because of the signed data agreement, as explained in Section 4.2.1.

The first step was normalization, where all letters were turned into lowercase. Then, the numbers were removed, as they did not provide any meaning to the LM performing the SA. Later, the contractions were expanded before the stop words were removed. Finally, the punctuation was removed, and the remaining terms were lemmatized. Lemmatization was selected instead of stemming because of the trade-off between time spent and quality. As the dataset size was not massive, the lemmatization step could be performed in a reasonable amount of time. After conducting these steps, the result was a preprocessed sentence.

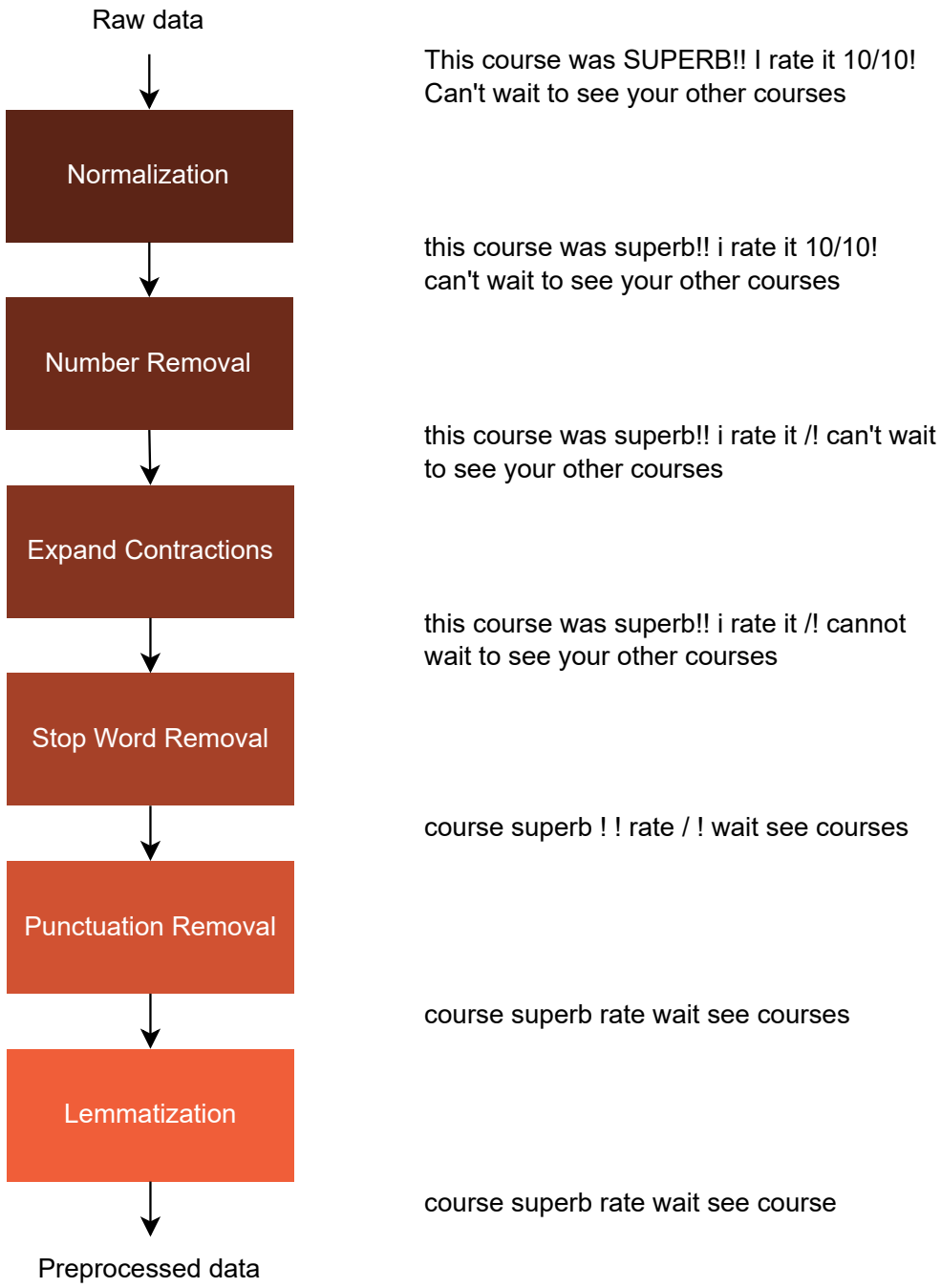


Figure 4.9: Data preprocessing steps with a fictional example on the right-hand side.

4.2.5 Data-Splitting Strategy

To evaluate an RS, the data must be split into a training and test set to make sure the same data is not used for tuning the parameters and for testing [14, p. 236]. The strategy for doing this is explained here.

The first step in the data-splitting strategy is to remove the reviews that do not contain a comment. This is done because the recommender model needs reviews that consist of both a rating and a comment during training. Also, for the users that have reviewed the same course multiple times, only the last review is used, based on the assumption that this is the most updated one. Hence, it has been made sure that there is a maximum of one review for each course per learner.

Next, a split ratio of 80/20, where 80% of the data is used in the training set and the remaining 20% in the test set, was chosen to have sufficient size for both splits. Since the recommender model aims to recommend items to users based on their and similar users' preferences, the training and test splits must contain reviews from all the users to be able to evaluate the recommendations for each user.

As the chosen split ratio is 80/20 (i.e., 4:1), all users that have given less than five reviews are ignored, as including these would not fulfill the minimum requirement of reviews for the split ratio. Then, for each of the users, 20% of their reviews are put into the test split, and the remaining 80% in the training split. For users with more than five reviews, every fifth review is put in the test set, while the four others are put in the training set, meaning the split ratio is not necessarily 80/20 for all users. By looking at Figure 4.8, one can see that 28k users have given a review with a comment to five or more courses. With a rough estimate, the total number of reviews is therefore 140k. Thus, the training and test splits contain 112k and 28k reviews, respectively.

As the comments are only used in the training set and not in the test set, the users who have given five reviews, where only four contain a comment, could also have been included. However, for simplicity reasons, only the users that have given five reviews or more containing comments were included. Comprising these as well would have increased the sizes of the training and test splits, but as they are sufficiently large, this was deemed unnecessary.

Chapter 5

Method

Three experiments were conducted to answer the three research questions proposed in Section 1.2. This chapter provides an introduction to the approaches followed to perform these. The motivation behind the architectural choices and a comparison to alternative decisions are also included.

First, the overall system architecture and approaches for all experiments are presented in Section 5.1. Then, the process followed to answer the first research question, i.e., Experiment 1, is elaborated on in Section 5.2, followed by the approaches followed to answer the second and third questions, i.e., Experiments 2 and 3 in Sections 5.3 and 5.4.

5.1 System Architecture

This section describes the architecture of the system used in the experiments. Figure 5.1 presents a systematic architecture overview. The yellow boxes related to data handling are included in all the experiments. The COCO dataset presented in Section 4.2 was used to generate recommendations and evaluate them. To make use of the course reviews in the dataset, they were extracted, preprocessed, and split into training and test sets, as described in Sections 4.2.4 and 4.2.5.

Further, the green boxes, related to rating handling, are relevant in Experiments 1 and 3. The process in these steps is further described in the sections related to these experiments, Sections 5.2 and 5.4. Lastly, the orange boxes are related to the recommendation process. The generation of recommendations differs for the experiments and will therefore be explained in each of the following sections. However, the evaluation of the RS was similar for all experiments.

Recommendation

As seen in Figure 5.1, the recommendation process comprises generation and evaluation. The following sections describe the generation process, but as the evaluation applies to all experiments, it is described here. After generating recommendations based on the training set, they are evaluated using the test set

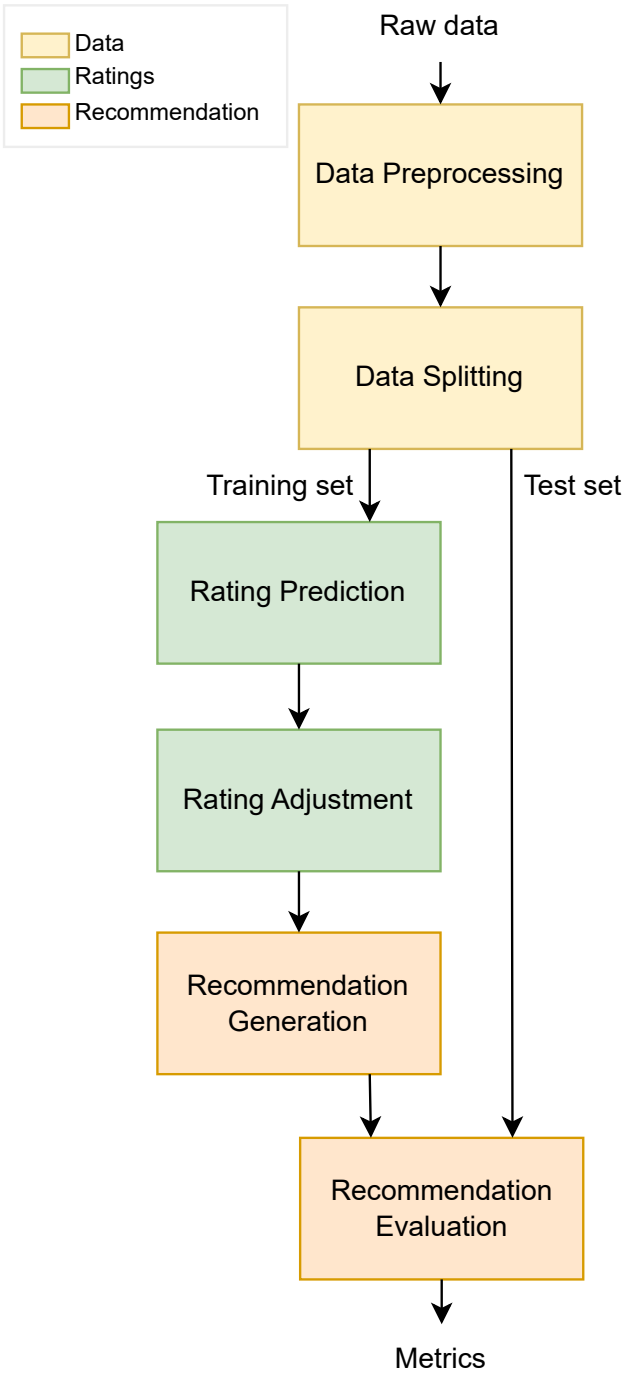


Figure 5.1: System architecture for the experiments. Yellow boxes represent steps related to the data, green boxes to ratings, and orange boxes to recommendations.

obtained through data splitting. As explained in Section 2.1.3, some evaluation metrics are needed to assess the system.

A common denominator for all the research questions is that the accuracy and ranking performance of the RS should be evaluated. To evaluate the accuracy of an RS, the two decision-support metrics *Precision* and *Recall* are often used. Therefore, these two metrics are used in this evaluation.

Further, some metrics are needed to evaluate the ranking accuracy of the RS. As explained in Section 2.1.3, nDCG is an expansion of DCG that measures the order in which items are recommended. Further, MAP looks at both the relevancy and order of the recommended items. Another ranking metric that was considered is Mean Reciprocal Rank (MRR). It measures where the first relevant item is, making it pretty simple to compute. However, as it does not evaluate the items after the first relevant one, it is not applicable here.

Based on these observations, the *Precision*, *Recall*, nDCG, and MAP metrics were used to evaluate the RS. These four metrics were described in Section 2.1.3.

5.2 Experiment 1: Adjustment of Ratings

As introduced in Section 1.2, the first research question is to figure out how the incorporation of sentiment from course reviews and adjustment of rating values based on sentiment affects the ranking accuracy of CR in MOOCs. In Section 3.3, some previous research on this topic was presented. This research question was inspired by the statement of Chen et al. [73] about how combining actual and inferred ratings would likely return better recommendations. Also, several other advantages of using reviews in RSs are presented by Al-Ghuribi and Mohd Noah [6]. Thus, a method to exploit information from the course reviews was needed.

As described in Section 3.3, SA is the most common approach to extracting information from course reviews. Meanwhile, several strategies could be used to exploit textual content. As this experiment aims to adjust the ratings based on course reviews, SA is suitable as it could be used to predict ratings based on the reviews' comments. When the ratings had been predicted, the original ratings had to be adjusted. To conduct this experiment, approaches were needed to perform SA on the course reviews and adjust the rating values based on this. Possible methods for these are discussed in Sections 5.2.1 and 5.2.2. Additionally, algorithms used to generate the recommendations are discussed in Section 5.2.3.

5.2.1 Rating Prediction Using Sentiment Analysis

The first step to answer the research question was to perform SA on the COCO course reviews to predict the ratings. The two well-known types of LMs introduced in Section 2.2.2, namely BERT and GPT, were taken into consideration for this task. A comparison of two BERT models and one GPT model follows.

BERT

As presented in Section 2.2.2, the BERT LM could be used to perform tasks such as SA. The version of BERT taken into consideration is the bert-base-multilingual-uncased-sentiment model[1] provided by Hugging Face[2]. It has been fine-tuned for SA on product reviews in six different languages, including English, Spanish, and others. This suits the task well, as the data in the COCO dataset is multilingual. However, the model is fine-tuned on product reviews, not course reviews.

As this model predicts a label between 1 and 5 stars, this could easily be translated to correspond to the ratings of 1-5 in the COCO dataset. Then, these predicted ratings could be used to adjust the original ratings. Further descriptions of how BERT works can be found in Section 2.2.2.

SiEBERT

Another LM based on BERT is the Sentiment in English Bidirectional Encoder Representations from Transformers (SiEBERT) model[3], introduced by Hartmann et al. [92]. It is specifically trained for the task of SA on 15 data sets with English data from diverse text sources. It outperforms models only trained on one type of text, such as the BERT model above.

The model predicts the sentiment of data as either positive or negative. This is a disadvantage compared to the BERT model, as the labels cannot directly be translated into ratings. Also, it is not trained in languages other than English, which is another disadvantage as the comments in the COCO dataset are in multiple languages.

GPT

A third possible LM is the GPT-3 model [36]. In comparison to BERT, it is autoregressive, while BERT is bidirectional, explained in Section 2.2.2. Because of this, BERT tends to perform better on tasks such as SA and Natural Language Understanding, as it bases its predictions on both left and right context. GPT models only consider the left context when predicting words in a sentence. As the task in this thesis is SA, the GPT model was not used.

Conclusion

Even though the SiEBERT model outperforms models trained only on one text type, it is unsuitable for multilingual data. Therefore, the multilingual BERT model was chosen as it seemed best suited as it handles texts in multiple languages well and predicts labels between 1 and 5 stars.

[1] https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment
[2] https://huggingface.co/
[3] https://huggingface.co/siebert/sentiment-roberta-large-english

5.2.2 Rating Adjustment Based on Predicted Ratings

After predicting the rating values based on the course reviews, the predictions were used to adjust the original ratings. As the original and predicted ratings were numerical values, these could be combined into one *adjusted* rating. Various research has used different approaches to combine actual and inferred ratings, as seen in Section 3.3.

The most common approach seen in the literature is to combine the ratings with different weights, as for example done by Osman [8]. This way, one can observe how different combinations of the two ratings affect the performance, and the best weights can be obtained. Based on these observations and previous literature, the original and predicted ratings were combined with different weights in this RS.

Some of the previous research also summed the ratings and took the average of them, such as Zhang et al. [11]. As this is the same as having the same weights for both ratings, equal weights were tested in this thesis. Several other approaches have been used to combine ratings and review texts, but as the data to combine both are numerical, the approach of using weights was chosen.

Specifically, the adjusted rating was calculated as seen in Equation (5.1), where W denotes the weight used. Given that W is between 0 and 1, the resulting adjusted rating will be on the interval $[0, 5]$, as both the original and predicted ratings are on this interval. Using the weight 0 results in 100% of the rating being decided by the original rating. In the opposite way, a weight of 1 results in the predicted rating. All values between 0 and 1 result in combinations of the original and predicted ratings, where a weight of 0.5 results in the average of the two.

$$\text{Adjusted rating} = W * \text{Predicted rating} + (1 - W) * \text{Original rating} \tag{5.1}$$

5.2.3 Algorithms to Generate Recommendations

To evaluate how the adjustment of the ratings affects the overall ranking and ordering of recommended MOOCs, an RS was implemented. Therefore, some algorithms had to be chosen to generate the recommendations, as shown in the first orange box in Figure 5.1. By selecting a set of algorithms, the impact of the adjusted ratings could be analyzed thoroughly.

A common algorithm to include for comparison is *Random*. As it recommends items randomly, all other algorithms should outperform it. Another type of algorithm is Content-Based Filtering (CBF), which recommends items based on their characteristics, as described in Section 2.1.1. As the recommender bases its recommendations on the adjusted ratings and no other characteristics, CBF algorithms were disregarded. Following, a few more recommendation algorithms are presented.

Popularity-Based

Other commonly used recommendation algorithms are popularity-based ones, such as Most Popular. As elaborated on in Section 2.1.1 these suggest items based on popularity. Thus, they can for example be used for new users on a platform to mitigate the cold start problem described in Section 2.1.2. Popularity-based algorithms are usually fast and low in complexity. Also, these rank items while recommending, which is desirable for the RS in this research. However, a downside is that these algorithms do not personalize recommendations, meaning the same items are recommended to all users.

Collaborative Filtering

As the reviews from each user are considered to generate recommendations, it is interesting to examine how personalized recommendations perform instead of generalized recommendations. A type of algorithm that considers each user's interests is CF, introduced in Section 2.1.1. These algorithms recommend items to users based on the interests of similar users or the similarity of items.

There exist many techniques used in CF, such as k-NN, MF, SVD, and Deep Learning. The latter is an interesting technique but was disregarded due to its complexity. A challenge for large datasets such as the COCO dataset is the sparsity problem, introduced in Section 2.1.2. SVD's dimensionality reduction can benefit large and sparse datasets. However, as Sarwar et al. [93] notes, SVD-based RSs provide limited scalability because of the computationally expensive MF step. Thus, this algorithm was ruled out as MOOC platforms tend to need scalability, as described in Section 2.1.2. However, there exist MF techniques that are less computationally expensive. These are advantageous through their handling of sparsity and scalability and providing personalized recommendations. However, they can struggle to handle the cold start problem and should be used in combination with other algorithms to mitigate this challenge. MF is further explained in Section 2.1.1. At last, k-NN algorithms also introduced in Section 2.1.1 are simple and flexible. They also struggle with scalability and data quality, as noise can impact its accuracy.

Conclusion

All in all, the algorithms above have advantages and disadvantages. With the COCO dataset and course reviews in mind, the algorithms considered were *Random*, *Most Popular*, and CF. The techniques used for CF were k-NN for its simplicity and flexibility and MF for its handling of sparsity and scalability. The diversity in the choice of algorithms laid a foundation for a thorough analysis of the impact of adjusted ratings.

5.3 Experiment 2: Rating-Based Approaches in Recommendation

As described in Sections 2.1.1 and 5.2.3, popularity-based recommendation algorithms can be used to recommend items to new users or users with few preferences. An essential aspect in popularity-based RSs is the rating-based approach used to find the most popular items, as presented in Section 3.4. The second research question provided in Section 1.2 revolves around comparing and analyzing the performance and ranking accuracy of a popularity-based RS using various rating-based approaches. Thus, the following five approaches were examined and analyzed with ranking accuracy in mind in this experiment.

5.3.1 Number of Ratings

A simple approach to finding the most popular course is to count its reviews. This approach is commonly used in the literature [80–86]. The most significant advantages of this approach are the calculation time and the simplicity. However, it only defines an item's popularity based on the number of reviews and does not consider what they contain. For example, if many users review a course poorly, it will be deemed popular with this approach. Even though many users have taken and reviewed the course, they were not happy about it, and it should not be considered popular. As this approach does not consider the rating values, it behaves equally with the original and adjusted ratings.

5.3.2 Sum of Ratings

Another approach is summing the ratings in the reviews of each course. This way, the ratings are considered when finding the most popular items. Generally, ratings are very positive [91], which also applies to the COCO dataset, as shown in Figure 4.2. Therefore, this approach was expected to have a similar result as counting the number of ratings. However, it was included as it looks at the rating values, which can also impact the recommendations based on the adjusted ratings.

5.3.3 Average of Ratings

A third approach is calculating each course's average rating, e.g., used in [80, 84]. This approach differs from the two above and focuses more on the rating values. Therefore, it benefits courses that have good ratings but have not been taken by many users. However, as introduced in Section 3.4, this approach also has some disadvantages.

Courses with fewer reviews will have a higher probability of being ranked high, a limitation introduced by Jannach et al. [80]. Say, for example, that course X has been reviewed by three users and course Y by 100 users. Both the courses have received a lot of positive reviews. X has received 100 reviews, where 99 have a rating of 5.0, and the last has a rating of 1.0. Thus, the average course

rating is 4.96. On the other hand, course *Y* has received three reviews, all rated as 5.0, giving an average rating of 5.0. By looking at the average rating to define popularity, course *Y* is more popular than course *X*. As the probability of a rating other than 5.0 increases when more reviews are given, this favors courses with fewer ratings. This is both an advantage and a disadvantage. On one side, new and undiscovered courses with potential can be exposed to more users. Conversely, courses taken and liked by many may be hidden.

5.3.4 Combination of Average and Summed Ratings

As explained, the approach of using average ratings has both advantages and disadvantages. Compared to the approaches of the number and sum of ratings, the average rating favors the rating values more. However, as observed, the average rating favors courses with fewer reviews. Though, as discussed earlier, the number of times a course has been reviewed could also represent popularity. As seen in Section 3.4, Rashid et al. [85] combined popularity-based and entropy-based techniques to get the advantages of both techniques. The disadvantages of both were attempted mitigated by adopting a similar approach by combining the approaches of average rating and the sum of ratings.

Combining these approaches could be done with different weights for exploration. However, in this research, the approaches were only weighted equally because of time limitations. To combine the approaches, they must be on the same interval, e.g., [0, 5]. The average ratings were already on this form, but the summed ratings were not. Therefore, two ways to convert the summed ratings into this interval are proposed below.

One way of converting the summed ratings is to distribute them along the interval with the ratio between them intact. In other words, the ratings were distributed *unevenly*. Figure 5.2 shows a visualization of this conversion. First, each item's summed rating was divided by the maximum of all summed ratings to convert all ratings to the interval of [0, 1]. Then, the ratings were multiplied by the maximum value of the interval, which was five as the interval is [0, 5]. This calculation is shown in Equation (5.2). The ratios between them were kept by dividing the summed rating by the maximum summed rating. Therefore, the points in Figure 5.2 are equally located for the summed and unevenly converted ratings, but the interval and numerical values changes. This method favors the rating values as the ratios between the items are kept.

$$\text{Unevenly converted rating} = \frac{\text{Summed rating}}{\text{Maximum summed rating}} * \text{Max value of interval} \tag{5.2}$$

Another way is to *evenly* distribute the summed ratings without respecting the ratio between them. In other words, the interval of [0, 5] was split into as many parts as there were items. Then, the items were evenly placed in the interval based on their ranking value. Therefore, some ratings increased while others decreased,

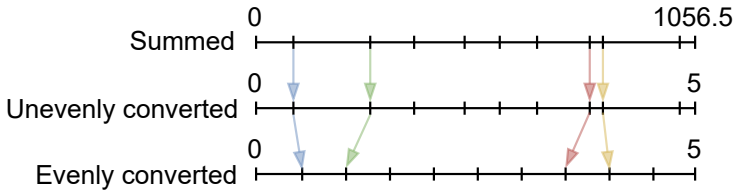


Figure 5.2: Conversion of summed ratings to another interval. The arrows show the conversion of selected points.

as observed in Figure 5.2. As this approach considered the ranking, which is based on the summed ratings but not the summed ratings themselves during placement, the ratios between the summed ratings were not kept. The first step was to calculate the distance between each item by dividing the max value of the interval by the number of items, as shown in Equation (5.3). Given that the items were ranked ascending by their summed ratings, the converted ratings were calculated by multiplying each item's rank by the distance, as in Equation (5.4).

$$\text{Distance} = \frac{\text{Max value of interval}}{\text{Number of items}} \tag{5.3}$$

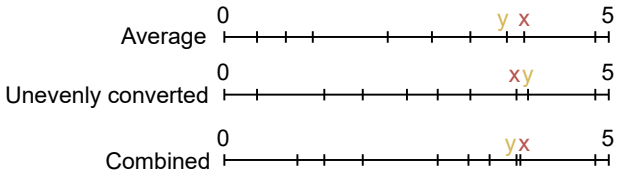
$$\text{Evenly converted rating} = \text{Item rank} * \text{Interval size} \tag{5.4}$$

The two rating values could be combined after converting the summed ratings to the same interval as the average ratings. The outcome of combining the values differed for the two approaches as the converted values varied, but the average ratings did not. An example of this combination is shown in Figure 5.3. The average ratings are given for two courses, x and y. Additionally, the unevenly and evenly converted ratings are shown in Figures 5.3a and 5.3b, respectively. The average and converted ratings were averaged to find the combined ratings. Therefore, as shown in Figure 5.3a, the combined rating for x is greater than the one for y for the unevenly converted ratings. However, for the evenly converted rating shown in Figure 5.3b, the combined rating for y is greater than for x. Thus, the combined ratings depend on which approach is followed.

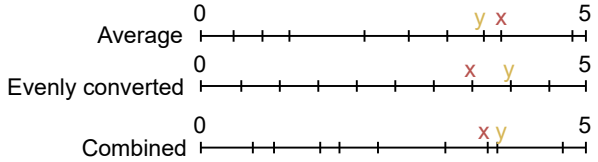
To sum up, the five rating-based approaches that were analyzed are *Number of ratings*, *Sum of ratings*, *Average rating*, *Combination of average and unevenly converted ratings*, and *Combination of average and evenly converted ratings*.

5.4 Experiment 3: Rating-Based Approaches With Adjusted Ratings

The third experiment was a combination of the two previous experiments. As formulated in Section 1.2, the third research question aims to examine how a



(a) For unevenly converted ratings.



(b) For evenly converted ratings.

Figure 5.3: Combining average rating with converted ratings.

popularity-based RS's performance and ranking accuracy with various rating-based approaches is affected by incorporating sentiment from course reviews. Thus, to answer this question, the third experiment expanded on adjusting ratings to improve recommendations. In contrast to Experiment 1, this experiment focused on the *Most Popular* recommendation algorithm presented in Section 5.2.3. More specifically, the performances of the rating-based approaches analyzed in Experiment 2 were compared. The motivations behind the two previous experiments were presented in Sections 5.2 and 5.3. This experiment was conducted to expand on the earlier experiments and more thoroughly analyze the performance of popularity-based RSs with adjusted ratings.

As with the other experiments, this experiment used the course reviews in the COCO dataset. The process was similar to the one for Experiment 1, visible in Figure 5.1. First, the data was gathered, preprocessed, and split into training and test sets. Then, the ratings were predicted using SA with BERT before the ratings were adjusted using the algorithm explained in Section 5.2.2. Then, the recommendations were generated using the five rating-based approaches presented in Section 5.3. Finally, the recommendations were evaluated using the metrics presented in Section 5.1.

Chapter 6

Experiments

As introduced in Chapter 5, three experiments were conducted in this work. This chapter provides a technical overview of the conducted experiments and related techniques. First, the experimental plan is described in Section 6.1 before the experimental setup is covered in detail with tools and parameters in Section 6.2.

6.1 Experimental Plan

The research aims to improve course recommendations as presented in Section 1.2. The three experiments described in Chapter 5 were conducted to do this. Experiments 1 and 3 had the same overall structure, while Experiment 2 differs as it did not include the rating prediction and adjustment process. Common to all experiments was the need for data, recommendation of courses, and evaluation of recommendations. The plan to conduct the experiments is presented in this section.

The first step of the experiments was to prepare the data. As described in Section 5.1, the course reviews had to be extracted, preprocessed, and split into training and test sets to be usable. Since the BERT model used for SA might perform better with stop words, as explained in Section 2.2.2, different versions of the dataset were created, with various preprocessing steps conducted. Then, the dataset versions were split into training and test splits following the strategy described in Section 4.2.5.

After having prepared the data, the plan differed between the experiments. For Experiments 1 and 3, the next step was performing SA on the course reviews in the training set. This process resulted in a predicted rating of 1-5 for each comment, which then was stored as a new attribute for each review. Subsequently, the original ratings had to be adjusted based on the predicted ratings. This was done using different weights described in Section 5.2.2. These steps were not conducted for Experiment 2, as it based its recommendations on the original ratings.

The last steps of the experiments were generating and evaluating the recommendations. Thus, an RS had to be implemented. For Experiment 1, the algorithms

presented in Section 5.2.3 were used to generate the recommendations. On the other hand, for Experiments 2 and 3, the *Most Popular* algorithm was used with the various rating-based approaches discussed in Section 5.3. For Experiment 2, the original ratings were used to generate recommendations, whereas Experiments 1 and 3 used the adjusted ratings. Finally, the recommendations were evaluated using the *Precision*, *Recall*, MAP, and nDCG metrics as described in Section 5.1.

6.2 Experimental Setup

The Python[1] programming language was used to conduct the experiments. This section describes the tools and parameters used to perform the experiments in Sections 6.2.1 and 6.2.2.

6.2.1 Tools

Below, the Python libraries used in the work are presented. The versions for each library are given inside the parentheses, and brief descriptions of the libraries and their usages are included.

Contractions[2] (0.1.73)	Handles contractions or combinations of words in a text. This thesis used it to expand contractions as a preprocessing step to standardize the text.
huggingface-hub[3] (0.10.1)	Makes ML models available to the public and provides lots of tools to set up and test these quickly. It was used to perform SA on the comments in the COCO dataset.
LensKit[4] (0.14.2)	Is a toolkit by Dr. Michael Ekstrand [94] for recommender experiments that has built-in recommender algorithms, evaluation techniques, and metrics. It was used to implement the recommendation algorithms and generate recommendations in this thesis.
Matplotlib[5] (3.7.1)	Provides tools to create visualizations of data. It was used to analyze and gather statistics about the dataset and present results.
NumPy[6] (1.23.4)	Provides tools to perform mathematical operations on arrays. In this thesis, it was used to handle NaN values.

[1] https://www.python.org/
[2] https://pypi.org/project/contractions/
[3] https://huggingface.co/
[4] https://lenskit.org/
[5] https://matplotlib.org/
[6] https://numpy.org/

nltk[7] (3.5)	Is a Natural Language ToolKit for python. It was used to preprocess the data through the usage of the *stopwords*, *punkt*, and *wordnet* modules. The former removes stop words, while the second was used for tokenization and the latter for lemmatization.
pandas[8] (1.0.5)	Is a data analysis and manipulation tool. In this thesis, it was used to handle operations on the COCO dataset efficiently and cleanly.
RanX[9] (0.3.8)	Is a library for ranking evaluation metrics in Python. It was used to evaluate this thesis' RS.

6.2.2 Implementation Details

Some parameters must be specified to run the RS. Some are specific to the algorithms in the LensKit library, while others are specific to the program. The recommender was tested with different parameters to find the optimal ones. In this section, the approach to finding these is presented, along with a description of the implementation in detail, focusing on the parameters.

Recommender Model

The LensKit Python library is the successor to the Java-based LensKit project[10], which was developed from 2012 to 2016. According to their research page[11], it has been used in 39 papers. The library was used to implement the RS because it provides various recommendation algorithms. Bałchanowski and Boryczka [95] inspired the implementation of the recommender model through their development of an RS using LensKit.

The algorithms used for recommendation in this thesis, *PopScore*, *Random*, *UserkNN*, *ItemkNN*, and *ALSImplicitMF* were predefined[12] in LensKit. These were chosen based on the reasoning presented in Section 5.2.3. The *ALSImplicitMF* algorithm is based on MF and ALS, which were introduced in Section 2.1.1. The *PopScore* algorithm is LensKit's version of the *Most Popular* algorithm. To enable different rating-based approaches for the popularity-based recommender in Experiment 2, the evaluation function of *PopScore* was overridden and implemented manually for each approach. However, for Experiments 1 and 3, the approach of *Sum of ratings* was used.

[7] https://www.nltk.org/index.html
[8] https://pandas.pydata.org/
[9] https://amenra.github.io/ranx/
[10] https://github.com/lenskit/lenskit
[11] https://lenskit.org/research
[12] https://lkpy.readthedocs.io/en/stable/algorithms.html

Some parameters had to be specified for the k-NN and *ALSImplicitMF* algorithms. For the k-NN algorithms, the *nnbrs*, *feedback*, *center*, and *aggregate* parameters were specified. *nnbrs* is the maximum number of neighbors used to score each item, i.e., *k* as described in Section 2.1.1. The recommender model was run with different values for both *ItemkNN* and *UserkNN* to find the value of k that provided the best results. Then, the results were evaluated using nDCG and MAP to decide the value for *k*.

The *feedback* parameter was set to explicit, as the data type in COCO is explicit. As with *nnbrs*, the *center* and *aggregate* parameters were tested to find the best combination. The former represents whether the rating vectors were normalized before the computation of similarities and aggregation of user rating values and could either be set to true or false. The latter represents the aggregation method and could be "sum" or "weighted average". The values for these parameters were also tested for both *UserkNN* and *ItemkNN*. Beyond these parameters, the default values of the k-NN algorithms in LensKit were used.

Further, some parameters for the *ALSImplicitMF* algorithm were defined. First, the *use_ratings* parameter was set to *True* so that the dataset's ratings were considered during prediction. The other specified parameter was the number of *features*. As with *k* for k-NN, different values for the parameter were tested. Otherwise, LensKit's default parameters for the *ALSImplicitMF* algorithm were used.

Sentiment Analysis

To perform the SA on the comments, a multilingual uncased BERT model from Hugging Face was used, as described in Section 5.2.1. It had been fine-tuned for the task of SA on product reviews in multiple languages. The task for the classifier was set to *sentiment analysis* for the best possible performance. Additionally, comments were truncated, or shortened, into sequences of 256 characters to enable conversion to fixed-size tensors[13]. These were specified by the *truncation* and *max_length* parameters in Hugging Face.

Evaluation

The RS was evaluated with metrics from the RanX library. The LensKit library also provides evaluation metrics, but as the MAP metric is not provided, the RanX library was used. In addition to MAP, it provides precision, recall, and nDCG. One limitation of the library is that it only accepts integers for the test set. Therefore, the ratings were converted from floating points to integers during the evaluation.

Program-Specific Parameters

When running the program, three parameters must be provided. The first one is the number of recommendations to generate for each user. The recommender

[13]https://huggingface.co/docs/transformers/pad_truncation

was tested with 10, 100, and 1000 recommendations to analyze its performance. These tests were performed on the ratings in the original COCO dataset for each of the five implemented algorithms used in Experiments 1 and 3.

Further, three versions of the dataset exist, each preprocessed by a different number of steps from the ones presented in Section 4.2.4. The second parameter is which dataset version to run the experiment on. This is represented by one of *all*, *WOstop_words*, or *WO_lemmatization*. The three alternatives represent all preprocessing steps, all steps except stop word removal, and all steps except stop word removal and lemmatization. The recommender was tested on each dataset version to find the best one. To obtain the baseline results to compare with in Experiments 1 and 3, the RS based its recommendations on the original values. Therefore, only the numerical values were considered, not the reviews. Thus, it did not make sense to analyze the performance of the RS for the baseline, as which preprocessing steps were performed would not impact the recommendations. Hence, the recommender was tested on the three dataset versions with adjusted ratings.

Finally, the third parameter is the experiment type. This could either be *baseline* to run the recommender on the original ratings or *sa_sentences* to run it on the adjusted ratings. Hence, to get the baseline results for Experiments 1 and 3, the *baseline* parameter must be provided. To get the results for the adjusted ratings, the *sa_sentences* parameter must be provided. To obtain the results for Experiment 2, the parameter for experiment type is *baseline*, as it is based on the original ratings.

Chapter 7

Results

The experiments conducted in this thesis were described in Chapter 5, and their implementation details in Chapter 6. The results obtained through the experiments are presented in this chapter. First, the results of the parametric testing described in Section 6.2.2 are displayed in Section 7.1. Then, the best parameters from these experiments are used to evaluate the recommender model and algorithms.

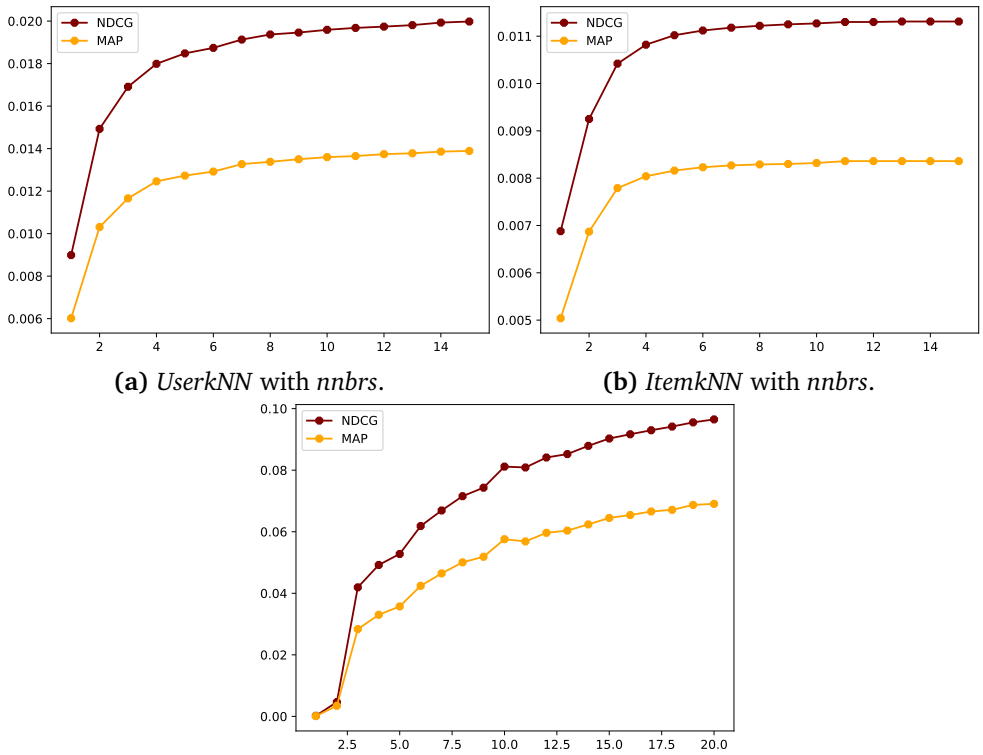
To evaluate the recommender model, the COCO dataset, presented in Section 4.2 was used. Further, the evaluation metrics used to test the RS were described in Section 5.1. The results of the three main experiments in this thesis are presented in Sections 7.2 to 7.4.

7.1 Parametric Testing

The RS was tested with different parameters to find the optimal ones, as described in Section 6.2.2. In this section, the results from these tests are shown. First, the parameters for the LensKit algorithms are shown in Section 7.1.1. Then, the program-specific parameters are shown in Sections 7.1.2 and 7.1.3.

7.1.1 Parameters for k-NN and ALSImplicitMF

Some parameters must be specified for the k-NN and *ALSImplicitMF* algorithms. All the tests were done using the nDCG and MAP metrics. For the k-NN algorithms, the recommender model's performance was evaluated with values of *nnbrs* from 0 to 15. The results for *UserkNN* and *ItemkNN* are shown in Figures 7.1a and 7.1b, respectively. As can be observed, the model's performance quickly improves before it stagnates around 6 *nnbrs* for both algorithms. For *ALSImplicitMF*, the *features* parameter was tested similarly, but with values between 0 and 20. The results from the test are visible in Figure 7.1c. The performance quickly improves until around ten features before increasing steadily.



(a) *UserkNN* with *nnbrs*. **(b)** *ItemkNN* with *nnbrs*.

(c) *ALSImplicitMF's* with *features*.

Figure 7.1: LensKit algorithms' performances with different values for specified parameters.

Further, the k-NN algorithms were tested with different values for *center* and *aggregate*. The results from these tests are displayed in Table 7.1. The best combination of the parameters was not normalizing the center and using sum as the aggregation mode. Thus, this was used in future experiments.

Table 7.1: The RS's performance with different values for parameters for the k-NN algorithms.

Center	Aggr.	Algorithm	NDCG	MAP
True	Sum	UserkNN	0.00054	0.00032
		ItemkNN	0.00062	0.00030
	WA	UserkNN	0.00029	0.00017
		ItemkNN	0.00126	0.00071
False	Sum	UserkNN	**0.00899**	**0.00602**
		ItemkNN	**0.00688**	**0.00504**
	WA	UserkNN	0.00175	0.00105
		ItemkNN	0.00222	0.00141

7.1.2 Number of Recommendations

The first program-specific parameter tested was the number of recommendations to generate for each user, as described in Section 6.2.2. The results of the experiments can be seen in Table 7.2.

Table 7.2: The RS's performance with different numbers of recommendations.

Algorithm	Num recs	NDCG	MAP	P	R
ALSImplicitMF	10	0.09789	0.07053	**0.02292**	0.16079
	100	0.15285	0.07969	0.00621	0.41461
	1000	**0.19244**	**0.08110**	0.00110	**0.70401**
UserkNN	10	0.01878	0.01297	**0.00471**	0.03184
	100	0.03578	0.01564	0.00169	0.11170
	1000	**0.05871**	**0.01633**	0.00044	**0.28475**
ItemkNN	10	0.01118	0.00827	**0.00249**	0.01750
	100	0.01660	0.00914	0.00064	0.04299
	1000	**0.03351**	**0.00952**	0.00028	**0.17633**
PopScore	10	0.01854	0.01103	**0.00531**	0.03643
	100	0.04422	0.01514	0.00231	0.15718
	1000	**0.08193**	**0.01626**	0.00068	**0.44294**
Random	10	0.00018	0.00009	0.00006	0.00039
	100	0.00100	0.00023	**0.00007**	0.00419
	1000	**0.00556**	**0.00026**	**0.00007**	**0.04117**

As can be observed, the results for nDCG, MAP, and *Recall* (R) increased for more significant numbers of recommendations. In contrast, the *Precision* (P) values decreased for larger numbers of recommendations. The one exception was the precision values for the *Random* algorithm, which were almost the same for all numbers of recommendations. In future experiments, the number of recommendations was set to ten because users would not want to search through tens or hundreds of recommendations.

7.1.3 Preprocessing Steps

The other program-specific parameter that was tested represents the preprocessed versions of the dataset to run the recommendations for, elaborated on in Section 6.2.2. The tests were run for the three versions of the dataset; all preprocessing steps, all steps except lemmatization, and all steps except lemmatization and stop word removal. These are labeled respectively in the table: *All*, *All - L*, and *All - LS*. The performance of the RS with the different preprocessing steps is shown in Table 7.3.

Table 7.3: The RS's performance with different preprocessing steps.

Algorithm	Preproc. steps	NDCG	MAP	P	R
ALSImplicitMF	All	**0.09888**	**0.07122**	**0.02315**	**0.16284**
	All - L	0.09631	0.06872	0.02284	0.16044
	All - LS	0.09635	0.06886	0.02269	0.15991
UserkNN	All	**0.01907**	**0.01324**	0.00473	**0.03215**
	All - L	0.01899	0.01316	0.00472	0.03206
	All - LS	0.01895	0.01312	**0.00474**	0.03205
ItemkNN	All	0.01103	0.00817	0.00246	0.01725
	All - L	0.01108	0.00820	0.00247	0.01732
	All - LS	**0.01117**	**0.00825**	**0.00249**	**0.01761**
PopScore	All	0.01863	0.01105	0.00535	0.03675
	All - L	0.01839	0.01096	0.00528	0.03596
	All - LS	**0.01866**	**0.01107**	**0.00536**	**0.03682**
Random	All	**0.00029**	**0.00017**	**0.00009**	**0.00060**
	All - L	0.00018	0.00009	0.00007	0.00035
	All - LS	0.00009	0.00005	0.00003	0.00023

As one can observe, the *ALSImplicitMF*, *UserkNN*, and *Random* algorithms performed best when all preprocessing steps were performed. There was one exception, as the *Precision* (P) values for *UserkNN* were not the best with all preprocessing steps, but the results were very close. *ItemkNN* and *PopScore*, on the other hand, performed best with all steps except lemmatization and stop word removal. In future experiments, the last dataset version was used.

7.2 Experiment 1: Adjustment of Ratings

Experiment 1 involves incorporating SA from course reviews and adjusting the rating values based on the predicted sentiment. It is described in more detail in Section 5.2. The results from this experiment are presented in this section.

7.2.1 Comparison of Rating Distributions

After having predicted and adjusted the original ratings, these were compared. One thing to note is that the rating values are on the same interval, i.e., [1, 5] but are represented differently. The original ratings are floating points that can end in .0 or .5, whereas the predicted ratings are represented as integers between one and five. As the adjusted ratings are combinations of these, they are floating points ending in .0, .25, .5, or .75. Therefore, the ratings were rounded to the nearest integer in this comparison. Also, only the ratings adjusted with the weight of 0.5,

i.e., the average between original and predicted ratings, are compared.

The comparison of the distributions of the original, predicted, and adjusted ratings is shown in Figure 7.2. Notice the significant difference in the number of predicted ratings of 5.0 compared to the original ones. However, there are twice as many predicted ratings of 4.0 than the original ones.

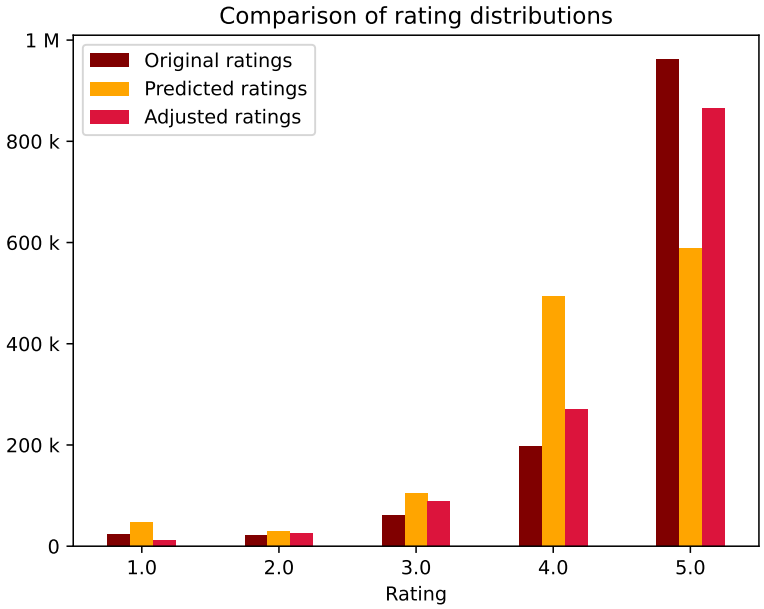


Figure 7.2: Original, predicted, and adjusted rating distributions.

7.2.2 Performance of Recommender With Adjusted Ratings

For Experiment 1, the RS was run on the adjusted ratings using different weights as described in Section 6.1. As the aim of the first research question was to explore how the RS's performance and ranking accuracy was affected by adjusting the ratings, the metrics *Precision* (P), *Recall* (R), nDCG and MAP were measured. Table 7.4 displays the results from the experiment.

The performance for each of the five algorithms is presented for each adjustment weight. The table's second column represents the weights used. A weight of 0 is the original rating, whereas 1 is the predicted rating. The three values between these, 0.25, 0.5, and 0.75, are adjustments of the original ratings based on the predicted sentiment. The closer to 0, the more the original rating weighs, and the closer to 1, the more the prediction weighs. A more in-depth description of this weighing can be found in Section 5.2.2.

Based on the results from the parametric tests in Section 7.1, the experiment was run with ten recommendations and all preprocessing steps except stop word removal and lemmatization. *ALSImplicitMF* and *UserkNN* performed best with a

weight of 1, with improvements of their nDCG scores of 1.18% and 1.54%, respectively. On the other hand, *ItemkNN* and *PopScore* performed best with adjusted ratings, with improvements of 0.36% and 0.65%, respectively. In other words, the improvements in performance were minor.

Table 7.4: Experiment 1: The RS's performance with adjusted ratings.

Algorithm	**Weight**	**NDCG**	**MAP**	**P**	**R**
ALSImplicitMF	0	0.09711	0.06978	0.02279	0.16031
	0.25	0.09732	0.06975	0.02286	0.16081
	0.5	0.09768	0.07023	0.02296	0.16108
	0.75	0.09663	0.06900	0.02286	0.16046
	1	**0.09826**	**0.07059**	**0.02304**	**0.16217**
UserkNN	0	0.01878	0.01297	0.00471	0.03184
	0.25	0.01892	0.01310	0.00474	0.03189
	0.5	0.01895	**0.01312**	0.00474	0.03205
	0.75	0.01898	**0.01312**	0.00475	0.03211
	1	**0.01907**	0.01311	**0.00482**	**0.03246**
ItemkNN	0	0.01118	0.00827	0.00249	0.01750
	0.25	**0.01122**	**0.00830**	**0.00250**	0.01760
	0.5	0.01117	0.00825	0.00249	**0.01761**
	0.75	0.01109	0.00815	**0.00250**	0.01758
	1	0.01108	0.00816	0.00249	0.01755
PopScore	0	0.01854	0.01103	0.00531	0.03643
	0.25	**0.01866**	**0.01107**	**0.00536**	**0.03682**
	0.5	**0.01866**	**0.01107**	**0.00536**	**0.03682**
	0.75	0.01842	0.01098	0.00529	0.03604
	1	0.01842	0.01098	0.00529	0.03604
Random	0	0.00024	**0.00014**	0.00008	0.00045
	0.25	0.00018	0.00009	0.00006	0.00040
	0.5	0.00025	**0.00014**	0.00009	0.00054
	0.75	0.00017	0.00008	0.00007	0.00033
	1	**0.00028**	0.00013	**0.00011**	**0.00055**

7.3 Experiment 2: Rating-Based Approaches in Recommendation

The aim of Experiment 2 was to analyze the impact different rating-based approaches have on the performance and ranking accuracy of a popularity-based RS. The recommender was run for the *PopScore* algorithm with the five approaches described in Section 5.3. Additionally, the *Random* algorithm was tested for comparison purposes. The results can be found in Table 7.5.

Note that the approach of *Combining the average and unevenly distributed ratings* performs the best for the nDCG, *Precision* (P), and *Recall* (R) metrics. However, for MAP, the approach of the *Number of ratings* performed best, with the previous approach following right behind. In addition, all the approaches except *Average rating* outperformed the *Random* algorithm.

Table 7.5: Experiment 2: The RS's performance with different rating-based approaches.

Rating-based approach	NDCG	MAP	P	R
Average Rating	0.00006	0.00004	0.00001	0.00011
Sum of Ratings	0.01854	0.01103	0.00531	0.03643
Number of Ratings	0.01860	**0.01110**	0.00531	0.03657
Avg and Uneven Distribution	**0.01868**	0.01107	**0.00535**	**0.03691**
Avg and Even Distribution	0.00642	0.00378	0.00195	0.01253
Random	0.00024	0.00014	0.00008	0.00045

7.4 Experiment 3: Rating-Based Approaches With Adjusted Ratings

The aim of Experiment 3 was to combine the two previous experiments, as described in Section 5.4. In other words, the rating-based approaches from Experiment 2 were tested on the adjusted ratings. Additionally, the *Random* approach was included for comparison. The results of the experiment are visible in Table 7.6. As for Experiment 1, the closer the weights are to 0, the more the original value weighs. Thus, the values for 0 in the table are the same as in Experiment 2, as seen in Table 7.5. A further description of the weight values can be found in Section 7.2.

Some results should be noticed. First, the *Number of ratings* approach had the same results for all weights. Second, this almost applied to the *Average rating* approach, except for the *Recall* (R) metric. Further, the *Sum of ratings*, *Combination of average and unevenly converted ratings*, and *Combination of average and evenly converted ratings* approaches differed for which weights they perform the best. They have in common to have the best results for weights between 0 and 0.5.

Table 7.6: Experiment 3: The RS's performance with different rating-based approaches on adjusted ratings.

Algorithm	Weight	NDCG	MAP	P	R
Average Rating	0	0.00006	0.00004	0.00001	0.00011
	0.25	0.00006	0.00004	0.00001	**0.00013**
	0.5	0.00006	0.00004	0.00001	**0.00013**
	0.75	0.00006	0.00004	0.00001	**0.00013**
	1	0.00006	0.00004	0.00001	**0.00013**
Sum of Ratings	0	0.01854	0.01103	0.00531	0.03643
	0.25	**0.01866**	**0.01107**	**0.00536**	**0.03682**
	0.5	**0.01866**	**0.01107**	**0.00536**	**0.03682**
	0.75	0.01842	0.01098	0.00529	0.03604
	1	0.01842	0.01098	0.00529	0.03604
Number of Ratings	0	0.01860	0.01110	0.00531	0.03657
	0.25	0.01860	0.01110	0.00531	0.03657
	0.5	0.01860	0.01110	0.00531	0.03657
	0.75	0.01860	0.01110	0.00531	0.03657
	1	0.01860	0.01110	0.00531	0.03657
Avg and Uneven	0	**0.01868**	**0.01107**	**0.00535**	**0.03691**
	0.25	**0.01868**	**0.01107**	**0.00535**	**0.03691**
	0.5	0.01836	0.01095	0.00526	0.03589
	0.75	0.01837	0.01095	0.00526	0.03589
	1	0.01823	0.01093	0.00518	0.03537
Avg and Even	0	0.00648	0.00378	**0.00195**	**0.01253**
	0.25	**0.00670**	**0.00480**	0.00158	0.01037
	0.5	0.00497	0.00274	0.00156	0.01039
	0.75	0.00447	0.00252	0.00138	0.00898
	1	0.00373	0.00196	0.00126	0.00820
Random	0	0.00024	**0.00014**	0.00008	0.00045
	0.25	0.00018	0.00009	0.00006	0.00040
	0.5	0.00025	**0.00014**	0.00009	0.00054
	0.75	0.00017	0.00008	0.00007	0.00033
	1	**0.00028**	0.00013	**0.00011**	**0.00055**

Chapter 8

Discussion

The current chapter discusses the results presented in Chapter 7. The discussions relate to the previous work presented in Chapter 3 and architectural choices made in Chapter 5. The chapter mainly follows the same structure as the results in the previous chapter. Therefore, the results of the parametric testing are discussed in Section 8.1. Later, the results from each experiment introduced in Chapter 5 are presented in Sections 8.2 to 8.4. Finally, some limitations and challenges are discussed in Section 8.5.

8.1 Parametric Testing

The recommender was tested with different parameters to find the optimal ones. The results from these tests were shown in Section 7.1. First, the results of the parametric tests related to the LensKit algorithms are discussed in Section 8.1.1 before the results associated with the program-specific parameters are discussed in Section 8.1.2.

8.1.1 Parameters for k-NN and ALSImplicitMF

The RS's performance using k-NN quickly improved before it stagnated around a k value of six, as seen in Figures 7.1a and 7.1b. As described in Section 2.1.1, k points were grouped into clusters that the recommendations were based on. Thus, low values of k would make the algorithm sensitive to noise and outliers in the data, decreasing performance. Further, a too large k would include irrelevant points in the clusters. Thus, the number of k chosen was six.

Using *ALSImplicitMF*, the algorithm also quickly improved until round ten *features*, but it continued to improve steadily afterward, as shown in Figure 7.1c. As this parameter refers to the dimensionality of the latent factors that represent users and items in the model, the RS's performance varied with different values. For low numbers of features, the model struggled to present the underlying patterns and preferences in the data accurately. Then, the performance was reduced as the model could not capture as many user-item relationships. On the other

hand, more features allowed the model to capture more intricate and nuanced patterns in the data. However, using too many features leads to overspecialization, explained in Section 2.1.2. Therefore, twenty features were used, as the model still improved until this value. Extensive tests were not done since the same configurations were used for baseline and further experiments, and the aim was to observe the impact of the adjusted ratings. Regardless of the number of features chosen, the *ALSImplicitMF* algorithm would obtain different results each time it was run, as it is based on randomness as explained in Section 2.1.1.

Even though the optimal parameters depend on the data, a comparison was made to another study using the LensKit library. The values used by Bałchanowski and Boryczka [95] for k were 23 for *UserkNN*, 44 for *ItemkNN*, and 21 for *features* in *ALSImplicitMF*. In this thesis, the value of six was chosen for k for both algorithms. For increased performance, different values for k for the two algorithms could have been tested, but by looking at the plotted results, the difference would have been minor. Regarding the value for *features*, their chosen values are larger than the optimal ones in this thesis, implying that they used a more intricate dataset.

8.1.2 Program-Specific Parameters

Some parameters must be provided when running the program, as elaborated on at the end of Section 6.2.2. Here, the results of the tests of the two program-specific parameters *Number of recommendations* and *Preprocessing steps* are discussed.

Number of Recommendations

The number of recommendations to generate for each user was tested with values of 10, 100, and 1000. The results of the tests can be found in Table 7.2. All the algorithms behaved similarly, except the *Random* algorithm's *Precision* scores. The RS's performances and ranking accuracies in these tests are discussed here.

As mentioned in Section 2.1.3, the nDCG metric considers both relevance and ranking position. When more recommendations are included in the evaluation, additional relevant items will likely be included in the list. Therefore, the nDCG values increased for larger numbers of recommendations. However, this increase would stagnate with more recommendations if the additional items are less relevant.

Increasing metric values for many recommendations also applied to the MAP metric. Including more recommendations would give a higher chance of capturing additional relevant items, which gives a higher average precision. As MAP is calculated by taking the average precision of each user and averaging these, it increases for higher parameter values. As with nDCG, the metric scores would stagnate at some point.

The *Recall* scores also increased for larger numbers of recommendations. This was expected as the metric is calculated by dividing the number of retrieved rel-

evant items by the total number of relevant items, as explained in Section 2.1.3. Thus, including more items in the evaluation would likely lead to the inclusion of more relevant items. However, an increased *Recall* score does not imply higher performance, as it does not consider the ranking or relevance of the recommendations.

The *Precision* metric represents the percentage of the relevant recommended items, as described in Section 2.1.3. In contrast to the other metrics, the scores decreased with more recommendations for the four first algorithms. This behavior shows that the RS successfully places relevant items higher in the recommendation list. For the *Random* algorithm, on the other hand, the scores were almost identical for all numbers of recommendations. This is because this algorithm randomly recommends items, and the relevant items will likely be distributed evenly throughout the list.

Experiments 1 and 3 compared the performance of the RS with original and adjusted ratings. However, the differences would likely be more significant with more recommendations based on the observations above. However, as users would not want to search through hundreds or thousands of recommendations, ten were considered in the experiments, as noted in Section 7.1.2.

Preprocessing Steps

As introduced in Section 2.2.1, human language has to be preprocessed for machines to understand it. Therefore, the RS was tested on three dataset versions. All versions included the first three preprocessing steps introduced in Section 4.2.4, normalization, number removal, and expansion of contractions. The first version also consisted of removing stop words and punctuation, and lemmatization. The second version included the three common steps and the two former steps, while the third version only included the three common steps and the removal of punctuation. Further descriptions of the versions and preprocessing steps were given in Sections 4.2.4 and 6.2.2. As can be observed in Table 7.3, the performance of the RS differed slightly using the different versions.

The hypothesis of the testing was that the rating predictions would be more accurate when stop words were included, as BERT is expected to perform better when stop words are preserved, as explained in Section 2.2.2. Therefore, the results for *All - LS* were expected to outperform the other dataset versions. This applied for *ItemkNN* and *PopScore*, but not for the remaining algorithms. Thus, which dataset version to use for optimal results depends on the algorithm.

Each recommendation algorithm was affected differently by the dataset versions. However, note that the version only excluding lemmatization never performed the best. Thus, this was disregarded when choosing which one to use in further experiments. For the two other versions, some algorithms performed better with all preprocessing steps, while others performed better without stop word removal and lemmatization. As there was no common pattern in the results, the *All - LS* version was chosen based on the hypothesis of BERT's performance.

8.2 Experiment 1: Adjustment of Ratings

As mentioned in Section 5.2, the first experiment aimed to answer the first research question, i.e., how incorporating SA of course reviews, and adjusting rating values based on sentiment, affects the performance and ranking accuracy of course recommendations in MOOCs. These two aspects and some trade-offs and considerations are discussed in this section.

8.2.1 Incorporation of Sentiment Analysis

The incorporation of SA in the recommendation was expected to improve the performance of the RS, based on previous research [8–13, 73]. Specifically, this hypothesis was based on the observation of Zheng et al. [7] that users explain their reasoning behind ratings through their reviews. Therefore, the different rating distributions in Figure 7.2 were wanted. If the distributions had been identical, incorporating sentiment would not have affected the recommendations. Instead, the predicted ratings were distributed more evenly along the five ratings than the original ones. Thus, incorporating sentiment helped to mitigate the challenge of distribution introduced in Section 4.2.3 by providing more diversity in the ratings.

Notice that the sum of original and predicted ratings of four and five was almost the same. Hence, many users give a rating of five, but based on their attached reviews, these were predicted to be ratings of four. This could result from how users rate items, in the sense that some users leave more positive ratings than others on average. Hence, this approach could also be used to normalize ratings. For ratings two and three, almost the same number was present for the three types of ratings. However, for one, there were more predicted ratings than the original ones, which can result from inadequate reviews that were wrongfully predicted.

8.2.2 Adjustment of Ratings Using Weights

In previous literature, actual and inferred ratings have been combined using different approaches, as described in Section 3.3. This thesis combined the original and predicted ratings using weights, elaborated on in Section 5.2.2. The results from the experiment can be seen in Table 7.4.

Generally, any rating adjustment outperformed the RS's performance with original ratings. The only exceptions were the nDCG and MAP of the *Random* algorithm. However, again, this algorithm randomly recommends items, so its recommendations were not affected by the ratings and can be ignored. For the remaining algorithms, which weight gave the best results differed. Based on this observation, using weights is a better approach than averaging, as done by for example Zhang et al. [11]. If the ratings had been combined with their average, these observations would not have been made, and the best results would not have been obtained.

By analyzing the results, one can observe that *ALSImplicitMF* and *UserkNN* performed best with predicted ratings (Weight $= 1$). Compared to using the original

ratings, their nDCG scores were improved by 1.18% and 1.54%, respectively, as noted in Section 7.2. However, the *ItemkNN* and *PopScore* algorithms performed best with weights of 0.25. The improvements for these two were minor compared to the original ratings, being 0.36% and 0.65%. These slight improvements when incorporating the sentiment differed for each algorithm because of how they work.

Li et al. [9] is the only research found using any of the same algorithms as in this thesis. They used different versions of ALS, introduced in Section 2.1.1, in their CF approach. However, they did not use the same metrics to evaluate their RS. Thus, the comparison is not valid but can pinpoint the expected behavior of the *ALSImplicitMF* algorithm. In their experiment, the ALS method with comprehensive ratings performed 5-12% better than the original ratings. This improvement is more significant than the one obtained here, but various factors can cause this, such as the data at hand. Comparisons to previous research for the other algorithms could not be made as no work was found using k-NN or popularity-based algorithms - in a reasonable amount of time.

As noted in Section 8.2.1, the predicted ratings were more evenly distributed along the five rating values. The adjusted ratings in Figure 7.2 are the average of the original and predicted ratings. By combining these ratings with their average, the rating distribution would naturally be normalized. As can be observed, the distribution of the adjusted ratings was more similar to the original ratings, with a preponderance of positive ratings. However, combining the actual ratings with the predicted ones made the ratings more spread out. The distributions of the adjusted ratings with different weights could have been included in the comparison but were not because the adjusted and predicted distributions represent the weights well enough.

8.2.3 Trade-offs and Considerations

The first consideration to note about Experiment 1 is regarding the evaluation. The impact of incorporating sentiment in course recommendations was measured using evaluation metrics. It would be interesting to conduct an online evaluation through user surveys or interviews, as described in Section 2.1.3, to explore how this incorporation affects user satisfaction and experience. Further, the users might have different expectations regarding the role of sentiment in recommendations. This and possible solutions, such as enabling or disabling sentiment in recommendations, could be looked into.

Another consideration is the time and resources needed to experiment. First, the number of recommendations discussed in Section 8.1.2 impacted the time required. Thus, even though the best results were obtained with 1000 recommendations, this was too time-consuming to do for all experiments. Second, the recommendation algorithms affected the necessary time and resources. The *Most popular* and *Random* algorithms were very fast, whereas the *ALSImplicitMF* and k-NN algorithms took more time during the recommendation. However, the differences were minor for smaller datasets and numbers of recommendations. How-

ever, they differed more with larger datasets, such as COCO. As the computational duration was some hours instead of days, this trade-off was taken to compare the RS's performance using different algorithms. Further, using original, predicted, or adjusted ratings did not affect the computational complexity, as they are all floating points on the same interval. However, the prediction step was time-consuming because SA is a resource-heavy process. Thus, the SA should be performed as few times as possible.

The implementation used a BERT model to predict ratings. As elaborated on in Section 5.2.1, several other LMs could have been used for this. Thus, interesting insights could be obtained by analyzing the prediction of ratings using another model. Similarly, there exist many recommendation algorithms. Using the same algorithms used by other researchers, the results could be more easily compared to other studies. Another aspect that affects the performance of the prediction is the quality of the data.

8.3 Experiment 2: Rating-Based Approaches in Recommendation

The second experiment aimed to analyze and compare the impact of different rating-based approaches in popularity-based RSs for MOOCs, as elaborated on in Section 5.3. To analyze the performance and ranking accuracy of the RS using different approaches, the nDCG, MAP, *Precision*, and *Recall* metrics were measured. The results from the experiment were presented in Section 7.3. The impact of different rating-based approaches and some trade-offs and considerations are discussed here.

8.3.1 Impact of Rating-Based Approaches

The three approaches *Sum of ratings*, *Number of ratings*, and *Combination of average and uneven distribution of ratings* performed the best. The two first would perform equally if all ratings were the same, as elaborated on in Section 5.3.2. As shown in Figure 4.2, the COCO dataset's rating distribution is pretty unbalanced with a preponderance of positive ratings. Therefore, these two approaches were expected to have quite similar results, but not identical. In fact, *Number of ratings* slightly outperformed *Sum of ratings* for all metrics except *Precision*, where they performed the same. The hypothesis for *Sum of ratings* was that it would outperform *Number of ratings* because considering the rating values would improve the recommendations, compared to only looking at the number of reviews given. However, this hypothesis is rejected because this performed worse than *Number of ratings*.

The *Average rating* approach was not expected to perform well because of the disadvantages described in Section 5.3.3. It was included for comparison as it has obtained adequate results in previous research [80, 84]. Although, here, the approach performed worse than *Random* for all metrics. However, combining

it with other approaches could improve these. With the hypothesis that *Sum of ratings* would outperform *Number of ratings* in mind, the former was combined with *Average rating* for the best possible results. Based on the observation that *Number of ratings* outperformed *Sum of ratings*, it could be interesting to test the combination of this with *Average rating*, in addition to the combination with *Sum of ratings*.

Two ways were used to convert the *Sum of ratings* to combine it with the *Average rating* approach, described in Section 5.3.4. When using the *Unevenly converted ratings*, the RS performed almost three times as well as with the *Evenly converted ratings* for all metrics. Based on the observation that *Sum of ratings* massively outperformed the *Average rating* approach when used individually, this is not surprising as the *Unevenly converted ratings* favors the summed ratings. Ultimately, combining the *Average rating* with *Unevenly converted ratings* performed the best.

8.3.2 Trade-offs and Considerations

Fundamentally, the advantage of popularity-based recommendation models is their simplicity, as described in Section 5.2.3. However, they have some disadvantages that restrict and limit the results. They solely rely on aggregating ratings and do not consider individual preferences. This lack of personalization can result in less relevant recommendations for individual users. Although, as explained in Section 2.1.1, they could effectively mitigate the cold start problem introduced in Section 2.1.2. Therefore, the usage of such algorithms should be based on the situation. For example, they can be used when new users enter the platform or to expose them to new courses. Usually, popularity-based recommendation algorithms are used with other algorithms, such as CF or CBF, for the best results. However, as the goal of this experiment was not to obtain the best possible results but rather to compare different rating-based approaches, this was not considered an option.

As the best rating-based approaches performed similarly in this experiment, they were analyzed beyond their performances. First, the computational complexity of the approaches differed. The *Number of ratings*, *Sum of ratings*, and *Average rating* approaches had the lowest complexities, as they count, sum, and calculate the mean of the ratings. The combined approaches were more complex because of the conversion of the summed ratings. Beyond this, they were similar to the former approaches as they combined the summed ratings with the *Average rating* by calculating the mean. Further, the time spent during the recommendation depends on the computational complexities. Finally, all these approaches recommended items quickly, and the time and resources needed to experiment were therefore not a problem.

8.4 Experiment 3: Rating-Based Approaches With Adjusted Ratings

Experiment 3 aimed to analyze how incorporating sentiment from course reviews affected a popularity-based recommender system's performance and ranking accuracy with various rating-based approaches. The results of the experiment were presented in Section 7.4.

The first thing to note is that the *Number of ratings* approach was not affected by adjusted ratings as it does not consider the ratings when recommending. Therefore, the results were the same for all weights for this approach. In retrospect, the method of Jannach et al. [80] of counting the number of positive ratings instead of all ratings could have been more useful in this experiment. However, as seen in Figure 7.2, the rating distributions of the original and adjusted ratings were pretty similar, so the changes would not have been massive. Although, the approach would have been affected by the adjusted ratings as opposed to the total number of ratings.

The *Average rating* approach was also unaffected by the adjusted ratings, except the *Recall* scores. This was expected as the distributions of the original and adjusted ratings were similar, making it likely that the average ratings would be similar. This also applied to the *Sum of ratings* approach since it was calculated by summing the rating values. Further, as in Experiment 1, the results for the *Random* algorithm were unaffected by the adjustment of the ratings because it bases its recommendations on randomness.

As experienced in Experiment 2, *Number of ratings* outperformed the *Sum of ratings* approach for the original ratings. Based on the hypothesis that considering the rating values would improve the recommendations, these were again compared. However, when basing the recommendations on the adjusted ratings, *Sum of ratings* outperformed *Number of ratings*. As the two approaches have very similar results for original and adjusted ratings, this improvement was minor but still present.

The combined approaches' performances did not improve significantly by incorporating sentiment. The *Combination of average and evenly converted ratings* approach performed worse using predicted ratings. Depending on the metrics, it performed best with a weight of 0.25 or 0 - being the original metrics. The *Combination of average and unevenly converted ratings* approach also performed best for these two weights. However, the performance was not that different using the other weights. Based on the observations about how the original and adjusted rating distributions affected the *Average rating* and *Sum of ratings* approaches, this lack of improvement was expected as these were the approaches combined.

Several studies incorporating sentiment in the literature are compared to popularity-based RSs. Still, no studies incorporating sentiment in the recommendation using these RSs could be found. Therefore, the results were not compared to any other research. To conclude, based on the results and the observations about the rating distributions, the performance and ranking accuracy of the RS was

not affected significantly by incorporating sentiment for different rating-based approaches.

8.5 Limitations and Challenges

Generally, the results for the different experiments were pretty bad compared to other studies. This section discusses limitations and challenges that could have impacted the results. First, the evaluation metrics used in the assessment are discussed in Section 8.5.1 before the experimental design and data in Sections 8.5.2 and 8.5.3.

8.5.1 Evaluation Metrics

The metrics used to evaluate the RS were nDCG, MAP, *Precition*, and *Recall*, as described in Section 5.1. These were selected as the research questions aimed to measure the performance and ranking accuracy of the RS. Other metrics that could have been used are, for example, MRR and MAE. As explained in Section 5.1, MRR was not applicable as it only focuses on the first relevant item. MAE, on the other hand, was disregarded because of its handling of missing values. This was not fitting as the COCO dataset used for evaluation is very sparse.

The choice of metrics resulted in some limitations. First, the selected metrics did not measure the diversity and novelty of the recommendations. As these are essential factors to consider when evaluating an RS, additional metrics, such as diversity measures or serendipity metrics, could have been included. Further, for Experiments 2 and 3, the results have not been compared to any related work. Not a lot of related work is present, but the results could have been compared more easily by including some of the metrics used in these other works.

Another limitation of these metrics was their lack of measuring user satisfaction. This is a common problem when evaluating an RS with offline feedback. In contrast, if online feedback was used in addition, user satisfaction and utility could have been measured as described in Section 2.1.3. However, due to time and resource constraints, this was disregarded.

Lastly, a limitation was that the metrics do not consider contextual factors such as time, location, or user preferences. Recommendations can vary depending on these factors, and the measured performance might not reflect future performance [14, p. 251]. The course reviews in the COCO dataset contain a timestamp, which could have been considered to weigh reviews differently.

8.5.2 Experimental Design

Some libraries used to develop and evaluate the RS were in early stages, as shown in Section 6.2.1. Specifically, this applied to the *Contractions*, *Hugging Face*, *LensKit*, and *RanX* libraries. The first one expanded contractions in reviews into combinations of words. Thus, it affected the SA, but ultimately, it would not

affect the results as these words do not contain much sentiment. The Hugging Face library provided the LM used to perform SA. The fact that the library was in its early stages did not matter, as the quality of the provided models is what was important. The utilized model had been downloaded almost 1M times over the previous month. Thus, the model was expected to perform well. However, the model's performance depended on the quality of the data.

The LensKit library was used to implement the RS, as described in Section 6.2.2. Getting started using LensKit was challenging as the library's documentation contained errors. Additionally, not many researchers that have used the library have their code publicly available. However, after coming across the research of Bałchanowski and Boryczka [95], implementing the RS became significantly more manageable. Thus, alternative libraries were not examined, as the library contained all the preferred recommendation algorithms. In retrospect, other libraries could have been investigated to compare them to LensKit.

An aspect to consider is that the LensKit and RanX libraries obtained different results when evaluating the RS. The differences were not groundbreaking but still present. This was also noted by Bałchanowski et al. in their code[1]. However, which library had the erroneous results is not known. To figure this out, using the Scikit-learn library[2] for evaluation was attempted. Though, this attempt was unsuccessful because of the format of the recommendations. As the results in LensKit and RanX were not too large, the attempt was canceled due to time constraints. However, the validity of the results was weakened because of this.

8.5.3 Data

The dataset used for evaluation can significantly impact the evaluation results and the assessment of the system's performance. After comparing existing datasets in the educational domain, the COCO dataset was chosen for evaluation, as described in Chapter 4. However, some aspects of the dataset are discussed here.

One aspect that affected the evaluation results is the data quality. The course reviews have been given by online users with diverse backgrounds related to languages, ages, genders, cultures, education, etc. This could impact the nature of how reviews are given and should be considered. As an example, the course reviews can be in any language. Thus, a multilingual BERT model was chosen instead of removing the multilingual data. However, a multilingual model might perform worse on English reviews than an LM trained explicitly on English data. In a possible extension of the experiments, this could have been investigated.

Some preprocessing was done on the reviews to make the LM able to perform SA. Still, typos were not handled, meaning some information could be misunderstood by the LM. Additionally, the removal of punctuation could result in unwanted behavior. For example, one of the reviews contained "cool..at". The periods were removed in the preprocessing, and the two words were merged into "coolat".

[1] https://github.com/mbalchanowski/RecRankAgg/blob/main/rec_rank/helpers/helpers.py
[2] https://scikit-learn.org/

The LM is not able to understand that the words should be split up, and the meaning of the word "cool" will not be included in the SA. Such problems result from the unstructured and difficult-to-handle human languages, decreasing the quality of the SA. For example, further steps should be taken to avoid such issues by adding spaces between the two words in the example.

Some reviews ended up empty after the preprocessing because of the stop word removal. These were dropped due to time constraints. This could affect the performance of the RS as there is less training data. However, as these reviews only contained stop words, one can argue that they would not be valuable in the SA. Alternatively, instead of dropping these reviews, the original ratings could have been used. Considering the research questions, this alternative is not applicable as the ratings could not have been adjusted.

After preprocessing the dataset, the reviews were split into training and test sets based on the strategy described in Section 4.2.5. Using this splitting strategy, 140k reviews were distributed between the two splits, as shown in Figure 4.8. Hence, a lot of the data is disregarded. Using another split ratio than 80/20, the RS could have been trained on more data. For example, a split ratio of 75/25 would have resulted in 90k additional reviews. However, this increase is not too significant compared to the total number of reviews in the dataset. The COCO dataset is pretty sparse, as most users have given one review, shown in Figure 4.8. If the model could have been trained on reviews only containing a rating and not necessarily a comment, the training set would have been much larger. Additionally, cross-fold validation should have validated the RS to ensure reliable results.

Based on the reflections above, extensive tests should have been done to determine why the RS performed poorly compared to other studies. To figure out if it was because of the data or libraries used in the implementation, the RS should have been tested using another dataset, such as a MovieLens dataset[3]. Then, the results obtained could have been compared to other research to determine why the results in this thesis were poor.

[3]https://grouplens.org/datasets/movielens/

Chapter 9

Conclusion and Future Work

The increasing amount of data on the Web has caused an overload problem for the users. As a result, Recommender Systems have been developed to reduce the time spent searching for content online. Several researchers have combined reviews with ratings to improve recommendations by using unused information in the reviews. Various approaches have been utilized to combine these, where the most common one is with weights. However, this is yet to be done in Course Recommendation.

This thesis aimed to improve Course Recommendation by focusing on three approaches. In the first approach, ratings of course reviews in the COCO dataset were predicted by performing Sentiment Analysis using a BERT model. Then, these predictions were combined with the original ratings using different adjustment weights. Following, a Recommender System with several recommendation algorithms was implemented and evaluated on the original, predicted, and adjusted ratings. The second approach compared different rating-based approaches' impact on the Course Recommendations. Then, these two approaches were combined into a third one, where various rating-based approaches were evaluated on the original, predicted, and adjusted ratings.

The remaining parts of this chapter will present the contributions of the thesis in Section 9.1 and some possible directions for future work in Section 9.2.

9.1 Summary of Contributions

Three research questions were proposed in Section 1.2 to accomplish the objectives of the thesis. Answering these research questions gives a summary of the contributions to the field.

RQ1 *How does incorporating sentiment analysis of course reviews, and adjusting rating values based on sentiment, affect the performance and ranking accuracy of course recommendations in MOOCs?*

An RS was implemented to analyze how its performance was affected by adjusting

ratings based on sentiment. In conclusion, all the compared algorithms performed better with adjusted than original ratings. The *ALSImplicitMF* algorithm's nDCG score was 1.54% better, while the remaining algorithms' improvements were smaller. The *UserkNN* and *ALSImplicitMF* algorithms performed best with predicted ratings, whereas *ItemkNN* and *PopScore* performed best with a weight of 0.25. In summary, incorporating SA and adjusting ratings based on sentiment improved the performance and ranking accuracy of course recommendations in MOOCs.

RQ2 *To what extent does the choice of rating-based approaches impact the performance and ranking accuracy of popularity-based recommender systems for course recommendation in MOOCs?*

Five approaches were implemented to analyze how different rating-based approaches impacted the performance of a popularity-based RS. Then, the recommendations generated by the RS for each approach were evaluated. To conclude, the three best-performing approaches performed about as well as each other, while the two others performed significantly worse. The approach *Combining the average and unevenly distributed ratings* performed the best, closely followed by *Number of ratings* and *Sum of ratings*. Which of these approaches being used did not impact the popularity-based RS significantly, but the two other approaches should be disregarded.

RQ3 *How is a popularity-based recommender system's performance and ranking accuracy with various rating-based approaches affected by incorporating sentiment from course reviews?*

The rating-based approaches above were evaluated on the adjusted ratings to analyze how various rating-based approaches were affected by incorporating sentiment from course reviews. To conclude, the approaches of the *Average rating* and *Number of ratings* were unaffected. In contrast, *Sum of ratings*, *Combination of average and unevenly converted ratings*, and *Combination of average and evenly converted ratings* had minor improvements compared to the original ratings. However, these improvements were so small they could be disregarded.

9.2 Future Work

Based on the results and findings of this research, several directions can be pursued to conduct further research on the topic.

An interesting approach to extending this work could be to explore other LMs for predicting labels and different ways of incorporating the sentiment into the ratings. Several fascinating approaches have been used in the literature, as presented in Section 3.3. As the COCO dataset is so sparse, evaluating the recommender model using another dataset could provide valuable insights. Additionally, other algorithms could be used in the recommendation or a hybrid system to see how

this is affected by adjusted ratings. Further, an extension to work related to the rating-based approaches in popularity-based RSs could be done. As mentioned, the *Number of ratings* approach is unaffected by the adjusted ratings. However, counting the number of positive ratings would affect this approach as it would be based on the rating values.

The primary source for the recommendation process in this work is a single-criterion, or overall, rating score. One future approach is extracting information about different course aspects through the reviews. Then, sub-ratings could be inferred through, for example, SA, similar to the ones present on TripAdvisor[1], where a restaurant's overall rating is divided into sub-ratings such as food, service, and value. Examples of possible categories for online courses are the course's difficulty and pace and the instructor's engagement. Based on the sub-ratings and user preferences, a multi-criteria RS could be developed to recommend courses. This would be quite a different way of combining ratings and reviews.

Another future approach is to perform more advanced SA on the reviews. Each review could be split into sentences or smaller parts and examined. Then, based on the content, they could be included or excluded in the SA. Because of the varying quality of the reviews, this approach could focus on exploiting the well-written reviews to improve the course recommendations. Thus, the time and resources needed to perform the SA would be lowered, meaning the resources could be used smarter by focusing on the well-written reviews. Optionally, this approach could be combined with the one above to focus on different aspects of the reviews.

Lastly, a possible future approach is to include more features from the dataset in the recommendation. For example, the COCO dataset contains much data about the courses and instructors that could be exploited. By examining the data, some features could be extracted and merged with the current work or used by themselves to improve the course recommendations.

[1]https://tripadvisor.com/

Bibliography

[1] B. Marr, *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read*, en, Section: Enterprise & Cloud. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/ (visited on 10/11/2022).

[2] M. C. Urdaneta-Ponte, A. Mendez-Zorrilla and I. Oleagordia-Ruiz, 'Recommendation Systems for Education: Systematic Review,' en, *Electronics*, vol. 10, no. 14, p. 1611, Jan. 2021, Number: 14 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2079-9292. DOI: 10.3390/electronics10141611. [Online]. Available: https://www.mdpi.com/2079-9292/10/14/1611 (visited on 12/09/2022).

[3] *A Decade of MOOCs: A Review of MOOC Stats and Trends in 2021*, en-US, Dec. 2021. [Online]. Available: https://www.classcentral.com/report/moocs-stats-and-trends-2021/ (visited on 28/11/2022).

[4] Y. Pang, N. Wang, Y. Zhang, Y. Jin, W. Ji and W. Tan, 'Prerequisite-related MOOC recommendation on learning path locating,' *Computational Social Networks*, vol. 6, no. 1, p. 7, Aug. 2019, ISSN: 2197-4314. DOI: 10.1186/s40649-019-0065-2. [Online]. Available: https://doi.org/10.1186/s40649-019-0065-2 (visited on 10/12/2022).

[5] C. Gütl, R. H. Rizzardini, V. Chang and M. Morales, 'Attrition in MOOC: Lessons Learned from Drop-Out Students,' en, in *Learning Technology for Education in Cloud. MOOC and Big Data*, L. Uden, J. Sinclair, Y.-H. Tao and D. Liberona, Eds., ser. Communications in Computer and Information Science, Cham: Springer International Publishing, 2014, pp. 37–48, ISBN: 978-3-319-10671-7. DOI: 10.1007/978-3-319-10671-7_4.

[6] S. M. Al-Ghuribi and S. A. Mohd Noah, 'Multi-Criteria Review-Based Recommender System–The State of the Art,' *IEEE Access*, vol. 7, pp. 169 446–169 468, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2954861.

[7] L. Zheng, V. Noroozi and P. S. Yu, 'Joint Deep Modeling of Users and Items Using Reviews for Recommendation,' in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17, New York, NY, USA: Association for Computing Machinery, Feb. 2017, pp. 425–

434, ISBN: 978-1-4503-4675-7. DOI: 10.1145/3018661.3018665. [Online]. Available: https://dl.acm.org/doi/10.1145/3018661.3018665 (visited on 18/04/2023).

[8] N. Osman, 'Contextual Sentiment Based Recommender System to Provide Recommendation in the Electronic Products Domain,' *International Journal of Machine Learning and Computing*, vol. 9, no. 4, Jul. 2020, ISSN: 20103700. DOI: 10.18178/ijmlc.2019.9.4.821.

[9] W. Li, X. Li, J. Deng, Y. Wang and J. Guo, 'Sentiment based multi-index integrated scoring method to improve the accuracy of recommender system,' en, *Expert Systems with Applications*, vol. 179, p. 115 105, Oct. 2021, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115105. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421005467 (visited on 30/05/2023).

[10] Y. Wang, M. Wang and W. Xu, 'A Sentiment-Enhanced Hybrid Recommender System for Movie Recommendation: A Big Data Analytics Framework,' en, *Wireless Communications and Mobile Computing*, vol. 2018, e8263704, Mar. 2018, Publisher: Hindawi, ISSN: 1530-8669. DOI: 10.1155/2018/8263704. [Online]. Available: https://www.hindawi.com/journals/wcmc/2018/8263704/ (visited on 23/02/2023).

[11] W. Zhang, G. Ding, L. Chen, C. Li and C. Zhang, 'Generating virtual ratings from chinese reviews to augment online recommendations,' *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 1, 9:1–9:17, Feb. 2013, ISSN: 2157-6904. DOI: 10.1145/2414425.2414434. [Online]. Available: https://dl.acm.org/doi/10.1145/2414425.2414434 (visited on 18/04/2023).

[12] C. Lee, D. Han, K. Han and M. Yi, 'Improving Graph-Based Movie Recommender System Using Cinematic Experience,' en, *Applied Sciences*, vol. 12, no. 3, p. 1493, Jan. 2022, Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app12031493. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1493 (visited on 23/02/2023).

[13] G. Ling, M. R. Lyu and I. King, 'Ratings meet reviews, a combined approach to recommend,' in *Proceedings of the 8th ACM Conference on Recommender systems*, ser. RecSys '14, New York, NY, USA: Association for Computing Machinery, Oct. 2014, pp. 105–112, ISBN: 978-1-4503-2668-1. DOI: 10.1145/2645710.2645728. [Online]. Available: https://dl.acm.org/doi/10.1145/2645710.2645728 (visited on 18/04/2023).

[14] C. Aggarwal, *Recommender Systems*. Jan. 2016, ISBN: 978-3-319-29657-9. DOI: 10.1007/978-3-319-29659-3.

[15] X. Su and T. M. Khoshgoftaar, 'A Survey of Collaborative Filtering Techniques,' en, *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, Oct. 2009, ISSN: 1687-7470, 1687-7489. DOI: 10.1155/2009/421425. [Online]. Available: https://www.hindawi.com/journals/aai/2009/421425/ (visited on 13/12/2022).

[16] E. Fix and J. Hodges, *Discriminatory analysis: nonparametric discrimination, consistency properties*, English. USAF school of Aviation Medicine, Feb. 1951, vol. 1.

[17] T. Cover and P. Hart, 'Nearest neighbor pattern classification,' *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, Conference Name: IEEE Transactions on Information Theory, ISSN: 1557-9654. DOI: 10.1109/TIT.1967.1053964.

[18] S. Funk, *Netflix Update: Try This at Home*. [Online]. Available: https://sifter.org/~simon/journal/20061211.html (visited on 22/05/2023).

[19] J. Bennett and S. Lanning, 'NetflixPrize-description.pdf,' en, *KDD Cup and Workshop*, Aug. 2007. [Online]. Available: https://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf (visited on 12/04/2023).

[20] Y. Koren, R. Bell and C. Volinsky, 'Matrix Factorization Techniques for Recommender Systems,' *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, Conference Name: Computer, ISSN: 1558-0814. DOI: 10.1109/MC.2009.263.

[21] M. Kuroda, Y. Mori and M. Iizuka, 'Initial Value Selection for the Alternating Least Squares Algorithm,' en, in *Advanced Studies in Classification and Data Science*, T. Imaizumi, A. Okada, S. Miyamoto, F. Sakaori, Y. Yamamoto and M. Vichi, Eds., ser. Studies in Classification, Data Analysis, and Knowledge Organization, Singapore: Springer, 2020, pp. 227–239, ISBN: 9789811533112. DOI: 10.1007/978-981-15-3311-2_18.

[22] G. Adomavicius and A. Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, Jun. 2005, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2005.99.

[23] D. M. Hawkins, 'The Problem of Overfitting,' *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, Jan. 2004, Publisher: American Chemical Society, ISSN: 0095-2338. DOI: 10.1021/ci0342472. [Online]. Available: https://doi.org/10.1021/ci0342472 (visited on 13/04/2023).

[24] D. C. Blair and M. E. Maron, 'An evaluation of retrieval effectiveness for a full-text document-retrieval system,' *Communications of the ACM*, vol. 28, no. 3, pp. 289–299, Mar. 1985, ISSN: 0001-0782. DOI: 10.1145/3166.3197. [Online]. Available: https://dl.acm.org/doi/10.1145/3166.3197 (visited on 03/05/2023).

[25] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl, 'Evaluating collaborative filtering recommender systems,' en, *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, Jan. 2004, ISSN: 1046-8188. DOI: 10.1145/963770.963772. [Online]. Available: https://dl.acm.org/doi/10.1145/963770.963772 (visited on 29/05/2023).

[26] K. Järvelin and J. Kekäläinen, 'Cumulated gain-based evaluation of IR techniques,' *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, Oct. 2002, ISSN: 1046-8188. DOI: 10.1145/582415.582418. [Online]. Available: https://dl.acm.org/doi/10.1145/582415.582418 (visited on 03/05/2023).

[27] G. Linden, B. Smith and J. York, 'Amazon.com recommendations: Item-to-item collaborative filtering,' *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003, Conference Name: IEEE Internet Computing, ISSN: 1941-0131. DOI: 10.1109/MIC.2003.1167344.

[28] *Netflix Research*, en. [Online]. Available: https://research.netflix.com/research-area/recommendations (visited on 05/05/2023).

[29] A. K. Uysal and S. Gunal, 'The impact of preprocessing on text classification,' en, *Information Processing & Management*, vol. 50, no. 1, pp. 104–112, Jan. 2014, ISSN: 0306-4573. DOI: 10.1016/j.ipm.2013.08.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457313000964 (visited on 05/05/2023).

[30] M. Toman, R. Tesar and K. Jezek, 'Influence of Word Normalization on Text Classification,' Jan. 2006. [Online]. Available: https://www.researchgate.net/publication/250030718_Influence_of_Word_Normalization_on_Text_Classification.

[31] C. Silva and B. Ribeiro, 'The importance of stop word removal on recall values in text categorization,' in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, ISSN: 1098-7576, vol. 3, Jul. 2003, 1661–1666 vol.3. DOI: 10.1109/IJCNN.2003.1223656.

[32] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv:1810.04805 [cs], May 2019. DOI: 10.48550/arXiv.1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 05/05/2023).

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, 'Attention is All you Need,' in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (visited on 05/05/2023).

[34] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, 'Improving language understanding by generative pre-training,' 2018, Publisher: OpenAI. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, 'Language Models are Unsupervised Multitask Learners,' en, 2019, Publisher: OpenAI. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[36] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry and A. Askell, 'Language models are few-shot learners,' *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020, Publisher: Curran Associates, Inc. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[37] OpenAI, *GPT-4 Technical Report*, arXiv:2303.08774 [cs], Mar. 2023. DOI: 10.48550/arXiv.2303.08774. [Online]. Available: http://arxiv.org/abs/2303.08774 (visited on 08/06/2023).

[38] B. Pang and L. Lee, 'Opinion Mining and Sentiment Analysis,' English, *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, Jul. 2008, Publisher: Now Publishers, Inc., ISSN: 1554-0669, 1554-0677. DOI: 10.1561/1500000011. [Online]. Available: https://www.nowpublishers.com/article/Details/INR-011 (visited on 05/05/2023).

[39] R. Kerin, M. Harvey and N. F. Crandall, 'Student Course Selection in a Non-Requirement Program: An Exploratory Study,' *The Journal of Educational Research*, vol. 68, no. 5, pp. 175–177, 1975, Publisher: Taylor & Francis, Ltd., ISSN: 0022-0671. [Online]. Available: https://www.jstor.org/stable/27536717 (visited on 09/12/2022).

[40] I. Ognjanovic, D. Gasevic and S. Dawson, 'Using institutional data to predict student course selections in higher education,' en, *The Internet and Higher Education*, vol. 29, pp. 49–62, Apr. 2016, ISSN: 1096-7516. DOI: 10.1016/j.iheduc.2015.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1096751615300087 (visited on 19/09/2022).

[41] I. D. Oladipo, J. B. Awotunde, M. AbdulRaheem, O. O. Ige, G. B. Balogun, A. R. Tomori and F. A. Taofeek-Ibrahim, 'An Improved Course Recommendation System Based on Historical Grade Data Using Logistic Regression,' en, in *Applied Informatics*, H. Florez and M. F. Pollo-Cattaneo, Eds., ser. Communications in Computer and Information Science, Cham: Springer International Publishing, 2021, pp. 207–221, ISBN: 978-3-030-89654-6. DOI: 10.1007/978-3-030-89654-6_15.

[42] N. D. Lynn and A. W. R. Emanuel, 'A review on Recommender Systems for course selection in higher education,' en, *IOP Conference Series: Materials Science and Engineering*, vol. 1098, no. 3, p. 032 039, Mar. 2021, Publisher: IOP Publishing, ISSN: 1757-899X. DOI: 10.1088/1757-899X/1098/3/032039. [Online]. Available: https://doi.org/10.1088/1757-899x/1098/3/032039 (visited on 12/09/2022).

[43] E. Babad and A. Tayeb, 'Experimental analysis of students' course selection,' en, *British Journal of Educational Psychology*, vol. 73, no. 3, pp. 373–393, 2003, ISSN: 2044-8279. DOI: 10.1348/000709903322275894. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1348/000709903322275894 (visited on 09/12/2022).

[44] Q. Zheng, L. Chen and D. Burgos, 'Certificate Authentication and Credit System of MOOCs in China,' en, in *The Development of MOOCs in China*, ser. Lecture Notes in Educational Technology, Q. Zheng, L. Chen and D. Burgos, Eds., Singapore: Springer, 2018, pp. 261–276, ISBN: 978-981-10-6586-6. DOI: 10.1007/978-981-10-6586-6_13. [Online]. Available: https://doi.org/10.1007/978-981-10-6586-6_13 (visited on 27/05/2023).

[45] Y. Hou, P. Zhou, J. Xu and D. O. Wu, 'Course recommendation of MOOC with big data support: A contextual online learning approach,' in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 106–111. DOI: 10.1109/INFCOMW.2018.8406936.

[46] Y. Zheng, R. Liu and J. Hou, 'The construction of high educational knowledge graph based on MOOC,' in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Dec. 2017, pp. 260–263. DOI: 10.1109/ITNEC.2017.8284984.

[47] N. Wan, X. Wu, S. Guo, L. Yang, Q. Han and R. Yin, 'Research on Personalized Recommendation of learning Resource Based on Big Data of Education,' in *2020 IEEE 2nd International Conference on Computer Science and Educational Informatization (CSEI)*, Jun. 2020, pp. 306–311. DOI: 10.1109/CSEI50228.2020.9142524.

[48] J.-P. Niyigena and Q. Jiang, 'A Hybrid Model for E-Learning Resources Recommendations in the Developing Countries,' in *Proceedings of the 2020 4th International Conference on Deep Learning Technologies*, ser. ICDLT '20, New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 21–

25, ISBN: 978-1-4503-7548-1. DOI: 10.1145/3417188.3417211. [Online]. Available: https://dl.acm.org/doi/10.1145/3417188.3417211 (visited on 27/05/2023).

[49] H. Hajri, Y. Bourda and F. Popineau, 'Personalized Recommendation of Open Educational Resources in MOOCs,' en, in *Computer Supported Education*, B. M. McLaren, R. Reilly, S. Zvacek and J. Uhomoibhi, Eds., ser. Communications in Computer and Information Science, Cham: Springer International Publishing, 2019, pp. 166–190, ISBN: 978-3-030-21151-6. DOI: 10.1007/978-3-030-21151-6_9.

[50] S. Sergis and D. G. Sampson, 'Enhancing Learning Object Recommendations for Teachers Using Adaptive Neighbor Selection,' in *2015 IEEE 15th International Conference on Advanced Learning Technologies*, ISSN: 2161-377X, Jul. 2015, pp. 391–393. DOI: 10.1109/ICALT.2015.50.

[51] M. Ceyhan, S. Okyay, Y. Kartal and N. Adar, 'The Prediction of Student Grades Using Collaborative Filtering in a Course Recommender System,' in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2021, pp. 177–181. DOI: 10.1109/ISMSIT52890.2021.9604562.

[52] K. Bhumichitr, S. Channarukul, N. Saejiem, R. Jiamthapthaksin and K. Nongpong, 'Recommender Systems for university elective course recommendation,' in *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2017, pp. 1–5. DOI: 10.1109/JCSSE.2017.8025933.

[53] N. A. Deraman, W. M. A. W. Ariffin, K. A. F. A. Samah, M. N. H. H. Jono, M. A. M. Isa, Z. F. Zamzuri and S. A. Yahaya, 'Covid19 and Higher Education: A Degree Course Recommender System Based on Personality Using Rule-Based,' en, *IOP Conference Series: Earth and Environmental Science*, vol. 704, no. 1, p. 012 021, Mar. 2021, Publisher: IOP Publishing, ISSN: 1755-1315. DOI: 10.1088/1755-1315/704/1/012021. [Online]. Available: https://doi.org/10.1088/1755-1315/704/1/012021 (visited on 28/09/2022).

[54] S. Channarukul, N. Saejiem, K. Bhumichitr, R. Jiamthapthaksin, V. Nicklamai and K. Terdvikran, 'Social-aware automated course planner: An integrated recommender system for university registration system,' in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Jun. 2017, pp. 545–548. DOI: 10.1109/ECTICon.2017.8096295.

[55] H. F. Unelsrød, 'Design and Evaluation of a Recommender System for Course Selection,' eng, *54*, 2011, Accepted: 2014-12-19T13:37:32Z Publisher: Institutt for datateknikk og informasjonsvitenskap. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/252564 (visited on 15/09/2022).

[56] L. Pappano, 'The Year of the MOOC,' en-US, *The New York Times*, Nov. 2012, ISSN: 0362-4331. [Online]. Available: https://www.nytimes.com/2012/11/04/education/edlife/massive-open-online-courses-are-multiplying-at-a-rapid-pace.html (visited on 27/05/2023).

[57] M. J. Gomez, M. Calderón, V. Sánchez, F. J. G. Clemente and J. A. Ruipérez-Valiente, 'Large scale analysis of open MOOC reviews to support learners' course selection,' en, *Expert Systems with Applications*, vol. 210, p. 118 400, Dec. 2022, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.118400. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422015081 (visited on 29/05/2023).

[58] Z. Zhao, Y. Yang, C. Li and L. Nie, 'GuessUNeed: Recommending Courses via Neural Attention Network and Course Prerequisite Relation Embeddings,' en, *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 16, no. 4, pp. 1–17, Nov. 2020, ISSN: 1551-6857, 1551-6865. DOI: 10.1145/3410441. [Online]. Available: https://dl.acm.org/doi/10.1145/3410441 (visited on 17/10/2022).

[59] H. Jain and Anika, 'Applying Data Mining Techniques for Generating MOOCs Recommendations on the Basis of Learners Online Activity,' in *2018 IEEE 6th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, Nov. 2018, pp. 6–13. DOI: 10.1109/MITE.2018.8747056.

[60] D. Yanhui, W. Dequan, Z. Yongxin and L. Lin, 'A Group Recommender System for Online Course Study,' in *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, Nov. 2015, pp. 318–320. DOI: 10.1109/ITME.2015.99.

[61] J. Zhang, B. Hao, B. Chen, C. Li, H. Chen and J. Sun, 'Hierarchical Reinforcement Learning for Course Recommendation in MOOCs,' en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 435–442, Jul. 2019, Number: 01, ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.3301435. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/3815 (visited on 25/10/2022).

[62] J. Qiu, J. Tang, T. X. Liu, J. Gong, C. Zhang, Q. Zhang and Y. Xue, 'Modeling and Predicting Learning Behavior in MOOCs,' in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ser. WSDM '16, New York, NY, USA: Association for Computing Machinery, Feb. 2016, pp. 93–102, ISBN: 978-1-4503-3716-8. DOI: 10.1145/2835776.2835842. [Online]. Available: https://dl.acm.org/doi/10.1145/2835776.2835842 (visited on 27/05/2023).

[63] H. Jung, Y. Jang, S. Kim and H. Kim, 'KPCR: Knowledge Graph Enhanced Personalized Course Recommendation,' en, in *AI 2021: Advances in Artificial Intelligence*, G. Long, X. Yu and S. Wang, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2022, pp. 739–750, ISBN: 978-3-030-97546-3. DOI: 10.1007/978-3-030-97546-3_60.

[64] Y. Zhao, W. Ma, Y. Jiang and J. Zhan, 'A MOOCs Recommender System Based on User's Knowledge Background,' en, in *Knowledge Science, Engineering and Management*, H. Qiu, C. Zhang, Z. Fei, M. Qiu and S.-Y. Kung, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2021, pp. 140–153, ISBN: 978-3-030-82136-4. DOI: 10.1007/978-3-030-82136-4_12.

[65] Y. Hou, P. Zhou, T. Wang, L. Yu, Y. Hu and D. Wu, *Context-Aware Online Learning for Course Recommendation of MOOC Big Data*, arXiv:1610.03147 [cs], Oct. 2016. DOI: 10.48550/arXiv.1610.03147. [Online]. Available: http://arxiv.org/abs/1610.03147 (visited on 08/10/2022).

[66] V. Garg and R. Tiwari, 'Hybrid massive open online course (MOOC) recommendation system using machine learning,' in *International Conference on Recent Trends in Engineering, Science & Technology - (ICRTEST 2016)*, Oct. 2016, pp. 1–5. DOI: 10.1049/cp.2016.1479.

[67] F. Wang, W. Pan and L. Chen, 'Recommendation for New Users with Partial Preferences by Integrating Product Reviews with Static Specifications,' en, in *User Modeling, Adaptation, and Personalization*, S. Carberry, S. Weibelzahl, A. Micarelli and G. Semeraro, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2013, pp. 281–288, ISBN: 978-3-642-38844-6. DOI: 10.1007/978-3-642-38844-6_24.

[68] C. W.-k. Leung, S. C.-f. Chan and F.-l. Chung, 'Integrating Collaborative Filtering and Sentiment Analysis: A Rating Inference Approach,' en, in *Proceedings of the ECAI 2006 workshop on recommender systems*, Aug. 2006, pp. 62–66. [Online]. Available: http://www.mysmu.edu/staff/caneleung/pub/rsw06_ratingInfer.pdf.

[69] W. Ma, M. Zhang, C. Wang, C. Luo, Y. Liu and S. Ma, 'Your Tweets Reveal What You Like: Introducing Cross-media Content Information into Multi-domain Recommendation,' en, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, Jul. 2018, pp. 3484–3490, ISBN: 978-0-9992411-2-7. DOI: 10.24963/ijcai.2018/484. [Online]. Available: https://www.ijcai.org/proceedings/2018/484 (visited on 23/02/2023).

[70] C.-C. Musat, Y. Liang and B. Faltings, 'Recommendation Using Textual Opinions,' en, *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2684–2690, 2013. [Online]. Available: https://www.ijcai.org/Proceedings/13/Papers/395.pdf.

[71] C. Yang, X. Yu, Y. Liu, Y. Nie and Y. Wang, 'Collaborative filtering with weighted opinion aspects,' en, *Neurocomputing*, vol. 210, pp. 185–196, Oct. 2016, ISSN: 09252312. DOI: 10.1016/j.neucom.2015.12.136. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231216305926 (visited on 29/05/2023).

[72] R. M. D'Addio, M. A. Domingues and M. G. Manzato, 'Exploiting feature extraction techniques on users' reviews for movies recommendation,' *Journal of the Brazilian Computer Society*, vol. 23, no. 1, p. 7, Jun. 2017, ISSN: 1678-4804. DOI: 10.1186/s13173-017-0057-8. [Online]. Available: https://doi.org/10.1186/s13173-017-0057-8 (visited on 29/05/2023).

[73] L. Chen, G. Chen and F. Wang, 'Recommender systems based on user reviews: The state of the art,' en, *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, pp. 99–154, Jun. 2015, ISSN: 1573-1391. DOI: 10.1007/s11257-015-9155-5. [Online]. Available: https://doi.org/10.1007/s11257-015-9155-5 (visited on 29/05/2023).

[74] J. McAuley and J. Leskovec, 'Hidden factors and hidden topics: Understanding rating dimensions with review text,' en, in *Proceedings of the 7th ACM conference on Recommender systems*, Hong Kong China: ACM, Oct. 2013, pp. 165–172, ISBN: 978-1-4503-2409-0. DOI: 10.1145/2507157.2507163. [Online]. Available: https://dl.acm.org/doi/10.1145/2507157.2507163 (visited on 30/05/2023).

[75] Y.-K. Ng and J. Linn, 'CrsRecs: A personalized course recommendation system for college students,' in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Aug. 2017, pp. 1–6. DOI: 10.1109/IISA.2017.8316368.

[76] M. J. Hazar, M. Zrigui and M. Maraoui, 'Learner comments-based Recommendation system,' en, *Procedia Computer Science*, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022, vol. 207, pp. 2000–2012, Jan. 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.09.259. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922011450 (visited on 29/05/2023).

[77] V. N. and A. K. K.m., 'E-learning course recommendation based on sentiment analysis using hybrid Elman similarity,' en, *Knowledge-Based Systems*, vol. 259, p. 110 086, Jan. 2023, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.110086. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122011820 (visited on 29/05/2023).

[78] S. K. Banbhrani, B. Xu, H. Lin and D. K. Sajnani, 'Taylor-ChOA: Taylor-Chimp Optimized Random Multimodal Deep Learning-Based Sentiment Classification Model for Course Recommendation,' en, *Mathematics*, vol. 10, no. 9, p. 1354, Jan. 2022, Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2227-7390. DOI: 10.3390/math10091354. [Online]. Available: https://www.mdpi.com/2227-7390/10/9/1354 (visited on 29/05/2023).

[79] R. Kumar and S. Bhatia, *Course Recommender System Architecture with Sentiment Score*, en, SSRN Scholarly Paper, Rochester, NY, Jul. 2021. DOI: 10.

2139/ssrn.3879643. [Online]. Available: https://papers.ssrn.com/abstract=3879643 (visited on 29/05/2023).

[80] D. Jannach, L. Lerche, I. Kamehkhosh and M. Jugovac, 'What recommenders recommend: An analysis of recommendation biases and possible countermeasures,' en, *User Modeling and User-Adapted Interaction*, vol. 25, no. 5, pp. 427–491, Dec. 2015, ISSN: 1573-1391. DOI: 10.1007/s11257-015-9165-3. [Online]. Available: https://doi.org/10.1007/s11257-015-9165-3 (visited on 07/06/2023).

[81] P. Cremonesi, Y. Koren and R. Turrin, 'Performance of recommender algorithms on top-n recommendation tasks,' in *Proceedings of the fourth ACM conference on Recommender systems*, ser. RecSys '10, New York, NY, USA: Association for Computing Machinery, Sep. 2010, pp. 39–46, ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721. [Online]. Available: https://dl.acm.org/doi/10.1145/1864708.1864721 (visited on 07/06/2023).

[82] R. Cañamares and P. Castells, 'Exploring social network effects on popularity biases in recommender systems,' en, *RSWeb@ RecSys*, 2014.

[83] P. Rakshit, S. Saha, A. Chatterjee, S. Mistri, S. Das and G. Dhar, 'A Popularity-Based Recommendation System Using Machine Learning,' en, in *Machine Learning in Information and Communication Technology*, H. K. Deva Sarma, V. Piuri and A. K. Pujari, Eds., ser. Lecture Notes in Networks and Systems, Singapore: Springer Nature, 2023, pp. 143–150, ISBN: 978-981-19509-0-2. DOI: 10.1007/978-981-19-5090-2_14.

[84] R. Cañamares and P. Castells, 'Should I Follow the Crowd? A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems,' in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18, New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 415–424, ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3210014. [Online]. Available: https://dl.acm.org/doi/10.1145/3209978.3210014 (visited on 07/06/2023).

[85] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan and J. Riedl, 'Getting to know you: Learning new user preferences in recommender systems,' in *Proceedings of the 7th international conference on Intelligent user interfaces*, ser. IUI '02, New York, NY, USA: Association for Computing Machinery, Jan. 2002, pp. 127–134, ISBN: 978-1-58113-459-9. DOI: 10.1145/502716.502737. [Online]. Available: https://dl.acm.org/doi/10.1145/502716.502737 (visited on 07/06/2023).

[86] A. Bellogín, P. Castells and I. Cantador, 'Statistical biases in Information Retrieval metrics for recommender systems,' en, *Information Retrieval Journal*, vol. 20, no. 6, pp. 606–634, Dec. 2017, ISSN: 1573-7659. DOI: 10.1007/s10791-017-9312-z. [Online]. Available: https://doi.org/10.1007/s10791-017-9312-z (visited on 07/06/2023).

[87] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim and J. Heo, 'EdNet: A Large-Scale Hierarchical Dataset in Education,' en, in *Artificial Intelligence in Education*, I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin and E. Millán, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 69–73, ISBN: 978-3-030-52240-7. DOI: 10.1007/978-3-030-52240-7_13.

[88] J. Kuzilek, M. Hlosta and Z. Zdrahal, 'Open University Learning Analytics dataset,' en, *Scientific Data*, vol. 4, no. 1, p. 170 171, Nov. 2017, Number: 1 Publisher: Nature Publishing Group, ISSN: 2052-4463. DOI: 10.1038/sdata.2017.171. [Online]. Available: https://www.nature.com/articles/sdata2017171 (visited on 02/12/2022).

[89] Skillmapper, *Did you pay for an online course and never finish it? You aren't alone. – SkillMapper Blog*, en. [Online]. Available: https://blog.skillmapper.com/did-you-pay-for-an-online-course-and-never-finish-it-you-arent-alone-and-lack-of-time-isnt-always-to-blame/ (visited on 05/06/2023).

[90] D. Dessì, G. Fenu, M. Marras and D. Reforgiato Recupero, 'COCO: Semantic-Enriched Collection of Online Courses at Scale with Experimental Use Cases,' en, in *Trends and Advances in Information Systems and Technologies*, Á. Rocha, H. Adeli, L. P. Reis and S. Costanzo, Eds., ser. Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2018, pp. 1386–1396, ISBN: 978-3-319-77712-2. DOI: 10.1007/978-3-319-77712-2_133.

[91] V. Schoenmueller, O. Netzer and F. Stahl, 'The Polarity of Online Reviews: Prevalence, Drivers and Implications,' en, *Journal of Marketing Research*, vol. 57, no. 5, pp. 853–877, Oct. 2020, Publisher: SAGE Publications Inc, ISSN: 0022-2437. DOI: 10.1177/0022243720941832. [Online]. Available: https://doi.org/10.1177/0022243720941832 (visited on 20/05/2023).

[92] J. Hartmann, M. Heitmann, C. Siebert and C. Schamp, 'More than a Feeling: Accuracy and Application of Sentiment Analysis,' en, *International Journal of Research in Marketing*, vol. 40, no. 1, pp. 75–87, Mar. 2023, ISSN: 0167-8116. DOI: 10.1016/j.ijresmar.2022.05.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167811622000477 (visited on 02/05/2023).

[93] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, 'Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems,' en, 2002.

[94] M. D. Ekstrand, 'LensKit for Python: Next-Generation Software for Recommender Systems Experiments,' in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20, New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 2999–3006, ISBN: 978-1-4503-6859-9. DOI: 10.1145/3340531.3412778. [On-

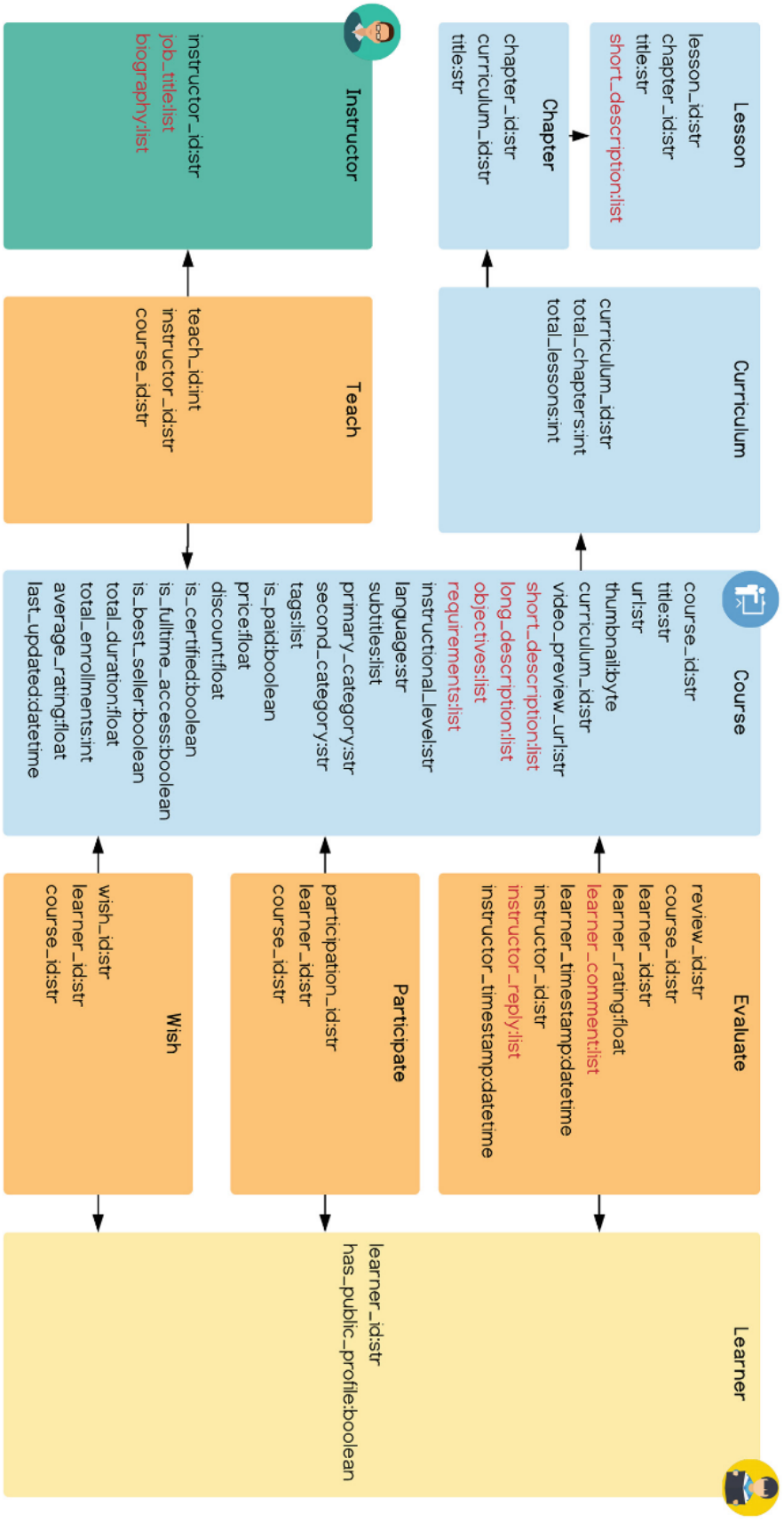
line]. Available: https://dl.acm.org/doi/10.1145/3340531.3412778 (visited on 03/05/2023).

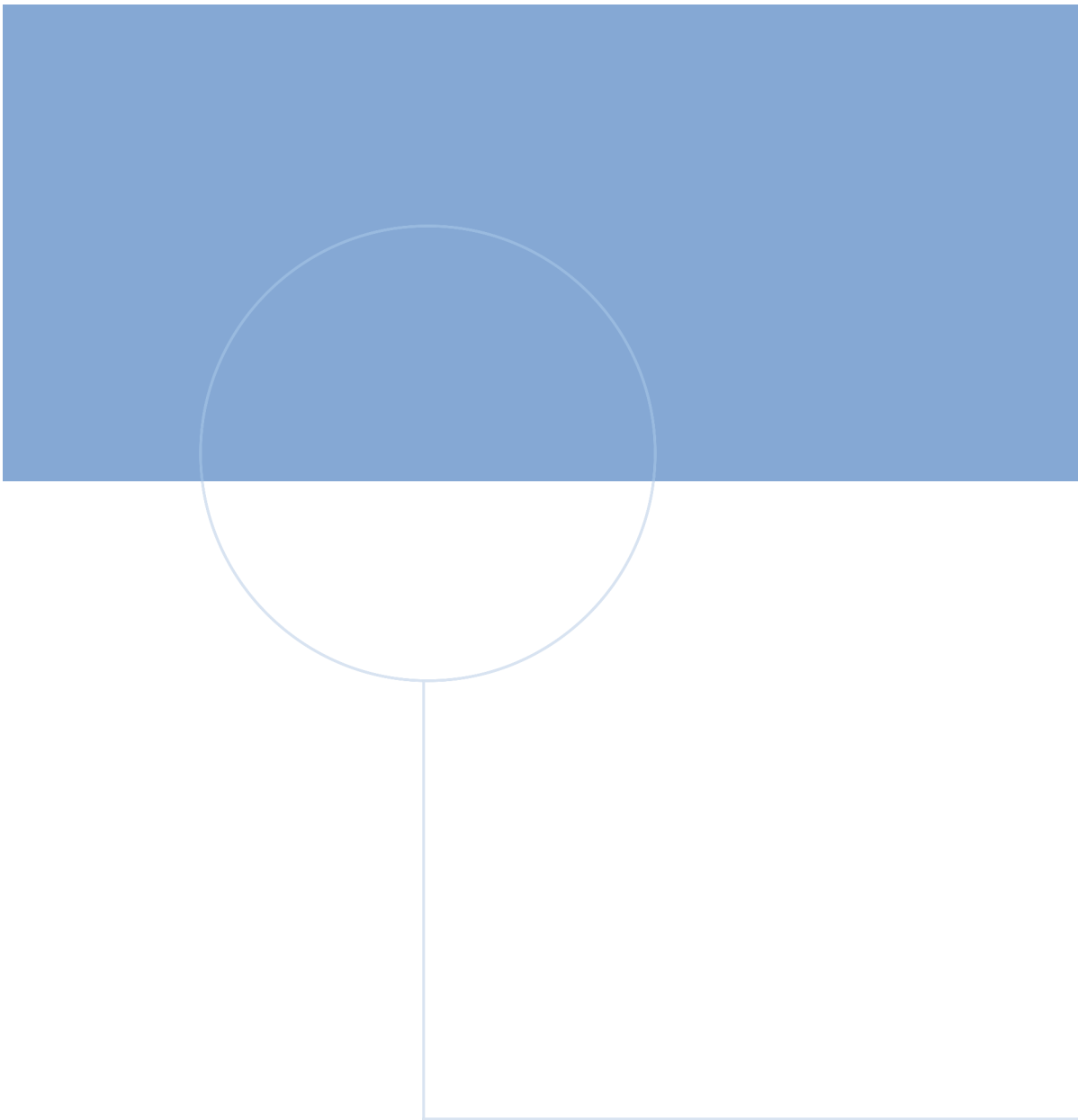
[95] M. Bałchanowski and U. Boryczka, 'A Comparative Study of Rank Aggregation Methods in Recommendation Systems,' en, *Entropy*, vol. 25, no. 1, p. 132, Jan. 2023, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1099-4300. DOI: 10.3390/e25010132. [Online]. Available: https://www.mdpi.com/1099-4300/25/1/132 (visited on 03/05/2023).

Appendix A

COCO Dataset Structure

The structure of the COCO dataset is present in this appendix. It contains different tables in which the data is included and the attributes in each table.


Lesson
lesson_id:str
chapter_id:str
title:str
short_description:list
Chapter
chapter_id:str
curriculum_id:str
title:str
Curriculum
curriculum_id:str
total_chapters:int
total_lessons:int
Instructor
instructor_id:str
job_title:list
biography:list
Teach
teach_id:int
instructor_id:str
course_id:str
Course
course_id:str
title:str
url:str
thumbnail:byte
curriculum_id:str
video_preview_url:str
short_description:list
long_description:list
objectives:list
requirements:list
instructional_level:str
language:str
subtitles:list
primary_category:str
second_category:str
tags:list
is_paid:boolean
price:float
discount:float
is_certified:boolean
is_fulltime_access:boolean
is_best_seller:boolean
total_duration:float
total_enrollments:int
average_rating:float
last_updated:datetime
Evaluate
review_id:str
course_id:str
learner_id:str
learner_rating:float
learner_comment:list
learner_timestamp:datetime
instructor_id:str
instructor_reply:list
instructor_timestamp:datetime
Participate
participation_id:str
learner_id:str
course_id:str
Wish
wish_id:str
learner_id:str
course_id:str
Learner
learner_id:str
has_public_profile:boolean



NTNU

Norwegian University of
Science and Technology