UNIVERSITY OF COPENHAGEN
DEPARTMENT OF COMPUTER SCIENCE

PhD Thesis

Simone Borg Bruun

Learning Recommendations from Sparse and Heterogeneous Data

Supervisors: Christina Lioma and Maria Maistro

This thesis has been submitted to the PhD School of The Faculty of Science, University of Copenhagen on March 31, 2024.

Abstract

Recommender systems have gained great success in helping users to navigate online services with a large variety of items, such as movies, music, retail and news, by inferring the users' preferences from their feedback on items like ratings, views and purchases. Personalized recommendations are also highly relevant in domains with few items, like the insurance domain, to guide the users' decision-making of products that are complex because they have many components and complex pricing. The process of generating recommendations in such domains raises some serious challenges with sparse and heterogeneous data that are under-researched in the field of recommender systems. Thus this thesis studies the generation of personalized recommendations under the special characteristics of complex item domains. Existing methods do not apply to complex item domains, because of the following reasons. The methods rely on a large volume of user feedback on items (e.g., purchases), but in complex item domains, the user feedback is sparse. This is because there may be few different items, and the items are not interacted with very often. Moreover, existing methods infer users' preferences for movie actors, music genres and so forth, whereas users have needs rather than preferences for complex products that tend to be temporal as their lives change. Finally, in complex item domains, the user feedback is of multiple types and modalities, such as actions on the website and phone calls with service agents, that are challenging to handle for existing methods. This is because they assume all feedback to be interactions with items (e.g., ratings or purchases) and they can, for example, not deal with a modality being missing. In addition to generating recommendations, we further study the process of generating explanations of recommendations under the special characteristics of complex item domains. Explanations of recommendations try to address the problem of explaining why items are recommended to a user. It is required by law and relevant for many domains, including complex item domains, to increase the trust and transparency of complex product recommendations. In summary, this thesis contributes improved recommendation and explanation methods that are suited for the aforementioned challenges within complex item domains, with a particular focus on tailor-made solutions for sparse-data recommendations, temporal needs and heterogeneous feedback in recommender systems. Collectively these contributions advance the state-of-the-art in recommender systems for this particular domain in ways that are scientifically interesting, timely, and societally relevant.

Resumé

Anbefalingssystemer har opnået stor succes i at hjælpe brugere med at navigere i onlinetjenester med et stort udvalg af varer, såsom film, musik, detail og nyheder, ved at udlede brugernes præferencer ud fra deres feedback på varer som anmeldelser, visninger og køb. Personaliserede anbefalinger er også yderst relevante i domæner med få varer, såsom forsikringsdomænet, til at vejlede brugernes beslutningstagning af produkter, der er komplekse, fordi de har mange komponenter og en kompleks prissætning. Processen med at generere anbefalinger i sådanne domæner rejser nogle alvorlige udfordringer med sparsom og heterogen data, der er underrepræsenteret indenfor forskning i anbefalingssystemer. I denne afhandling studeres derfor genereringen af personlige anbefalinger under de særlige karakteristika i komplekse varedomæner. Eksisterende metoder passer ikke til komplekse varedomæner af følgende årsager. Metoderne er afhængige af en stor mængde brugerfeedback på varer (f.eks. køb), men i komplekse varedomæner er brugerfeedbacken sparsom. Dette skyldes, at der kan være få forskellige varer, og der bliver ikke interageret med varerne ret ofte. Desuden udleder eksisterende metoder brugernes præferencer for filmskuespillere, musikgenrer og lignende, hvorimod brugere har behov snarere end præferencer for komplekse produkter, der har tendens til at være tidsbestemte i takt med, at deres liv ændrer sig. Endelig kan brugerfeedbacken i komplekse varedomæner være flere typer og have forskellige modaliteter, såsom handlinger på hjemmesiden og telefonopkald med kunderådgivere, der er udfordrende at håndtere for eksisterende metoder. Det er fordi, de antager, at al feedback er interaktioner med varer (f.eks. anmeldelser eller køb), og de kan for eksempel ikke håndtere, at en modalitet mangler. Ud over at generere anbefalinger studerer vi yderligere processen med at generere forklaringer på anbefalinger under de særlige karakteristika i komplekse varedomæner. Forklaringer på anbefalinger forsøger at løse problemet med at forklare, hvorfor bestemte varer anbefales til en bruger. Det er lovpligtigt og relevant for mange domæner, herunder komplekse varedomæner, til at øge tilliden til og gennemsigtigheden af komplekse produktanbefalinger. Alt i alt bidrager denne afhandling med forbedrede anbefalings- og forklaringsmetoder, der er velegnede til de førnævnte udfordringer inden for komplekse varedomæner med særligt fokus på skræddersyede løsninger til datasparsomme anbefalinger, tidsmæssige behov og heterogen feedback i anbefalingssystemer. Tilsammen fremmer disse bidrag den nyeste viden inden for anbefalingssystemer for dette særlige domæne på måder, der er videnskabeligt interessante, aktuelle og samfundsrelevante.

List of Publications

This thesis includes the following papers as chapters, listed in the order of their appearance in the thesis (* denotes equal contribution).

1. Simone Borg Bruun*, Kacper Kenji Leśniak*, Mirko Biasini, Vittorio Carmignani, Panagiotis Filianos, Christina Lioma, and Maria Maistro. Graph-Based Recommendation for Sparse and Heterogeneous User Interactions. In *Advances in Information Retrieval*, ECIR '23, pages 182–199. Springer Nature Switzerland, 2023. URL `https://doi.org/10.1007/978-3-031-28244-7_12`

2. Simone Borg Bruun, Maria Maistro, and Christina Lioma. Learning Recommendations from User Actions in the Item-poor Insurance Domain. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pages 113–123. Association for Computing Machinery, 2022. URL `https://doi.org/10.1145/3523227.3546775`

3. Simone Borg Bruun, Christina Lioma, and Maria Maistro. Recommending Target Actions Outside Sessions in the Data-Poor Insurance Domain. *ACM Transactions on Recommer Systems*, 2023. URL `https://doi.org/10.1145/3606950`

4. Simone Borg Bruun, Krisztian Balog, and Maria Maistro. Dataset and Models for Item Recommendation Using Multi-Modal User Interactions. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24. Association for Computing Machinery, 2024 **- In Press.**

5. Simone Borg Bruun, Christina Lioma, and Maria Maistro. Feature Attribution Explanations of Session-based Recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24. Association for Computing Machinery, 2024 **- Submitted.**

The following list of papers were also published during this PhD study, but are not included in the thesis.

1. Simone Borg Bruun. Learning Dynamic Insurance Recommendations from Users' Click Sessions. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 860–863. Association for Computing Machinery, 2021. URL `https://doi.org/10.1145/3460231.3473900` **- Doctoral Symposium.**

2. Toine Bogers, Cataldo Musto, David Wang, Alexander Felfernig, Simone Borg Bruun, Giovanni Semeraro, and Yong Zheng. FinRec: The 3rd International Workshop on

Personalization & Recommender Systems in Financial Services. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 688–690. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3523227.3547420

Acknowledgements

Along with this PhD thesis, I have received invaluable support from many people.

First and foremost I would like to thank my supervisors Christina Lioma and Maria Maistro for giving me guidance, critical feedback and freedom to explore my research interests. They have challenged and inspired me with their extensive experience, competences and commitment to scientific research.

I would also like to thank the company, Alka, that was hosting this industrial PhD, and especially John Egebo from Alka for being open-minded and curious about scientific research. I am grateful for the guidance into the challenges of a real-world industry and the resources I have been provided for my research. In addition, I would like to thank my other colleagues in Alka for supporting me in navigating the company's databases, software, systems and domain-specific knowledge.

Next, I would like to thank all the collaborators I have been lucky to work with. Thanks to the people from the research community in Denmark on recommender systems and the information retrieval lab at the University of Copenhagen for interesting presentations, discussions and knowledge sharing on common research interests. I was also lucky to have a research stay at the University of Stavanger in Norway. Here the Information Access and Artificial Intelligence group led by Krisztian Balog included me in their engaging research environment which I am grateful for.

Lastly, I would like to thank my family and friends for cheering on my scientific work, for bearing with me in hectic times, and for distracting me with enjoyable experiences that filled me with energy and pleasure.

Table of Contents

Chapter 1

Executive Summary

1.1 Introduction

With the increased development of the World Wide Web, people are to a greater extent left to themselves when shopping, streaming, browsing and so forth, instead of being advised by, for example, a salesperson in a physical store. This has created the need for recommender systems that can help users of online services find relevant content by inferring their interests from data. Recommender systems have gained great success in large item domains such as the movie, music, retail, book and news domains, by using the users' preferences for genres, actors, colors, styles and similar, to ease the users' choice overload with personalized recommendations. In contrast, this thesis focuses on domains with complex items. Complex items are items with many components and where the pricing is based on complex factors such as risk and durability. Examples of such domains are banking, pension, insurance, real estate and automobiles. Like many other domains, customers of these domains also have an increased interest in online self-service, but sales activities of complex products require expert knowledge. Recommender systems can offer great benefits for complex item domains by either improving the efficiency of service agents or automatizing the decision-making process for the customers.

Although a recommender system is highly relevant for complex item domains, these domains have some special characteristics that raise new challenges in the research field of personalized recommendation models:

1. As opposed to the domains where recommender systems have primarily gained traction by easing the overload of items, the number of different items is typically very small in complex item domains. There may be only 15 possible items to interact with. For comparison, there are likely several thousands of movies, music tracks, books and similar in the traditional domains. In addition, users rarely buy complex products as they are usually bought for a long-term commitment (e.g., one year)

because their utility is not realized immediately. For example, the utility of a pension saving is not realized until you retire or the utility of an insurance product is not realized until you have an insurance claim (e.g., theft or accident). Both the small number of items and the low purchase frequency cause a **sparse amount of information** of the users over the items, as each user has not interacted with many items.

2. Lifestyle changes affect customers' need for complex products. For example, customers' housing and car needs may change when they have children. Therefore the recommendation task is much more a question of capturing **temporal needs** rather than preferences of, for example, movie actors and music genres that are more connected to previous preferences.

3. Although users' purchases of items are limited, various other user actions exist in complex item domains such as updating personal information and claims reporting in the insurance domain. Moreover, users interact with such domains through more than one channel like website and call center. Exploiting this **heterogeneous user behavior** is essential to understand users' needs for complex products and generate personalized recommendations.

Hence, this thesis studies the generation of personalized recommendations under sparse, temporal and heterogeneous conditions that simultaneously characterize complex item domains. Each condition is itself also relevant for many other domains such as sparsity for a start-up business. In addition to generating recommendations, explaining why an item is recommended to a user is an important topic within recommender systems because it increases the trust and transparency of the system and it is required under various legislations. We further study explanations of temporal recommendations as it is relevant for complex item domains where temporal recommendations are well-suited and explanations of complex product recommendations are important.

The remainder of this chapter is structured as follows. Section 1.2 introduces the context of this project as an industrial PhD as well as the insurance domain that we use as a focal point. Section 1.3 provides an introduction to recommender systems and the theoretical background for the challenges related to complex item domains. Section 1.4 gives a detailed overview of the contributions in the papers included as chapters in this thesis. Section 1.5 summarizes the contributions of this thesis and points to prospects for future work.

1.2 Industrial Context

This thesis is an industrial PhD carried out in collaboration with a Danish insurance company called Alka[1]. I have been employed in Alka in their department for mathematical modeling. The employment gave me access to Alka's databases and software as well as the other employees' expertise on data, systems, products, and processes like customer experience and user interface.

Alka was established in 1944 and deals with insurance for individuals and small businesses. The company approximately has insurance contracts with 350,000 customers and currently has a five percent market share in Denmark. The company serves its customers through its website, call center, chat and e-mail communication. Their strategy has a specific focus on promoting its digital channels including website, chat and e-mail.

The nature of the collaboration with Alka resulted in using the insurance domain as a recurrent example of a complex item domain in this thesis. This domain is therefore introduced next.

In the insurance domain, the items to be recommended are insurance products. An insurance product is a contract between the insurance company and the customer. For a payment in advance, the company takes the obligation to pay compensation to the customer if a loss is caused under the terms of policy. Common insurance products for individuals are car, accident and house insurance. Each insurance product typically has some additional coverages with extended benefits that are purchased separately from the base insurance product, for example, roadside assistance for car insurance or dental injury for accident insurance. Also, additional coverages can be recommendable items. Insurance products are complex as they comprise many components, like deductibles and liability limits, and the price is based on the complex risk of claims. Consequently, the insurance domain shares the challenges for recommendations in complex domains: there are few different insurance products, users rarely buy insurance products, users have needs rather than preferences of insurance products, and users have heterogeneous behavior on the insurance company's different channels like website and call center.

A recommender system in the insurance domain can serve different purposes. One purpose is to help new customers identify their needs when they contact the company for the first time. But when a life event occurs to a customer, it often means that this customer has to adjust insurance products. For instance, if you have a child, you might want accident insurance for that child or if you move far away from your job, you may need additional coverage for your car insurance. Therefore, recommender systems in this domain often focus on another purpose of insurance recommendations which is to help existing customers to continuously adjust their insurance products to suit their needs.

The collaboration with the company has given a particular insight into the insurance

[1] `https://www.alka.dk/`

domain and access to real-world data. Alka is experiencing many customers who are not properly insured because they have not adjusted their insurance products to new circumstances in their lives. It is most often discovered by insurance agents when they are speaking with a customer in another context (e.g., claims reporting). Therefore, a recommender system is relevant to help the customers who are primarily using the company's digital channels as well as to support the insurance agents.

The customers mainly interact with Alka through two channels: the company's website and call center. The company's website is divided into four sections: (1) on the e-commerce sections the users can buy insurance products and additional coverages; (2) on the claims reporting section the users can report claims (e.g., theft or car accident); (3) on the information section the users can find information about terms of policy, payment processes and so forth; (4) on the personal account section the users can change their personal information like employment (required if you have an accident insurance) and e-mail address. The users can do all the same actions over the phone by contacting the company's call center. In this thesis, we exploit all the different actions on the website and over the phone to learn recommendations.

At the start of this PhD program, the insurance company did not have a recommender system in operation. During the project, Alka had its first recommender system implemented which currently is in the process of being A/B-tested with real users. This recommender system is based on our contributions in Papers 2 and 3 (Chapters 3 and 4). Both on the website interface and in the call center, the user will be recommended up to three items. This is why we focus on the first three items when evaluating the recommender systems in our contributions.

This thesis makes use of data collected in Alka. The data was collected in the periods between October 1, 2018 to September 30, 2020 and May 1, 2022 to April 30, 2023. All the data is collected with consent from the individuals involved as well as stored in accordance with the legal legislation for data protection and privacy. Along the way, analyses have been provided to enhance the transparency of our models, including a fairness analysis in paper 3 (Chapter 4) where we investigate biases of protected demographic characteristics.

1.3 Theoretical Background

This section provides an introduction to the basic concepts in recommender systems as well as the theoretical background for specific challenges related to complex item domains, namely heterogeneous and multi-modal data, the sparsity problem, temporal recommendations and explanations of those. The papers included in this thesis are cross-referenced when relevant.

1.3.1 Introduction to Recommender Systems

The task of a recommender system is to suggest items relevant to a user [3]. That is, given a set of users and a set of items, the goal is to rank the top-k most relevant items for each user, where k is a small number compared to the total number of items (e.g., 10 or 20 out of thousands of items). This is usually done by estimating a score for each user-item pair (e.g., predicting ratings or purchase probabilities), and then ranking the items according to the score.

To solve this problem, we focus on the case where a recommendation model is used to automatically infer recommendations from data. The recommendation problem can also be solved by requesting the users to explicitly specify what they want, defined as *knowledge-based recommender systems* [69].

1.3.1.1 Key Properties of Feedback Data

A driving force for the development of recommendation models is the ease with which the World Wide Web enables users to provide feedback about content for example in terms of ratings or likes/dislikes [3]. Feedback can also be more implicit. If a user buys a product or listens to a music track it may be seen as a preference for that item. This type of feedback is even easier to collect on the World Wide Web than explicit feedback (e.g., ratings or likes), it comes at almost no cost and is available in large amounts [74]. Recommendation models are typically based on the previous feedback given by a user such that past interests are used as indicators of future choices. The feedback can take different forms depending on the application setting. It can be in the form of ratings, for instance, as numerical integer values from one to five, as binary feedback with two values corresponding to positive or negative feedback, or as unary feedback when the user can only specify a positive preference for an item. For instance, a purchase can be considered as positive feedback whereas not purchasing an item is not necessarily a dislike. Note that the users can provide feedback in non-numerical forms, like a star rating system, but it will usually be converted to a numerical scale when used in a recommendation model. In Section 1.3.2 we will cover challenges of feedback data like the heterogeneity and ambiguity in it, missing feedback and temporal changes in preferences.

1.3.1.2 Basic Approaches for Recommendation Models

The approaches for inferring recommendations from feedback are broadly categorized into two basic types: *collaborative filtering approaches* and *content-based approaches*. Elements from both can also be combined, which is then called *hybrid approaches*.

Collaborative filtering approaches use only the users' historical feedback such as ratings or purchases of items [124]. A user has typically given feedback on some items, these are items with observed feedback, while there are other items that the user has not yet

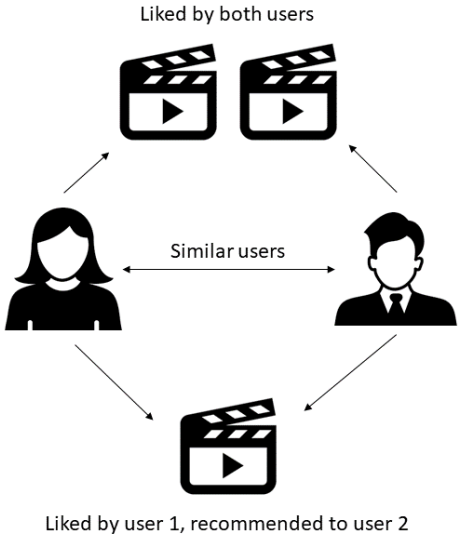
given feedback on, being items with unspecified feedback. Collaborative filtering uses the observed feedback to estimate a user's future feedback on the items that the user has not given feedback on yet. The items with the most positive estimated feedback (e.g., highest ratings or most purchases) are recommended to the user. The basic idea is that unspecified feedback can be estimated from the observed feedback because it is often highly correlated across various users and items. For example, if two users have given similar feedback on some items, the feedback on an item that is only specified by one of them is also likely to be similar. This similarity can be used to make inferences for the user who has not yet interacted with that item. See Figure 1.1 for an illustration of the concepts in collaborative filtering.

Content-based approaches use both descriptive features of the items and the users' feedback, but only the users' own feedback [4]. The descriptive features can be tabular features like actor, genre, color or unstructured features like textual descriptions or images of the items. The features of items that a user has interacted with are used to predict whether the user will like an item that the user has not yet interacted with. For example, a user who is interested in a movie within the history genre is more likely to be interested in another historical movie rather than in an action movie. See Figure 1.1 for an illustration of the concepts in content-based approaches.

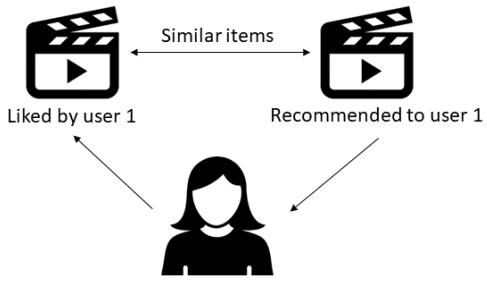
An advantage of collaborative filtering is that it does not require access to item features that are not always available. Additionally, the recommendations tend to be more novel and serendipitous, whereas content-based approaches tend to be overspecialized. This is because the recommended items do not necessarily share the same features as the user has previously liked when using collaborative filtering. An advantage of content-based approaches is that the recommendation for one user does not rely on other users' feedback, which is a problem when there is a sparse amount of feedback from other users. Unlike collaborative filtering, the content-based approach also has the advantage that it can recommend a new item that no one has previously interacted with. This is called the *cold start problem* [3]. In content-based approaches, a new item can be recommended because it shares the same features as the items that have previously received feedback from the user. In collaborative filtering, a new item needs to receive feedback from some users before it can be recommended. What applies to both approaches is that they need a certain amount of previous feedback from a user, to provide this user with high-quality recommendations.

1.3.1.3 Collaborative Filtering Methods

This thesis focuses on advancing collaborative filtering methods. In collaborative filtering, the feedback is usually represented in a user-item matrix with users as rows, items as columns and the feedback in the entries [124]. See Figure 1.2a for an example with unary feedback (i.e., the user can only specify a positive preference for an item). We want



(a) Collaborative filtering: Two users are similar because they have liked the same items. An item that is only observed (and liked) by user 1 is then recommended to user 2.



(b) Content-based: Two items are similar because they share the same features. User 1 has only observed (and liked) one of the items. The other item is then recommended to user 1.

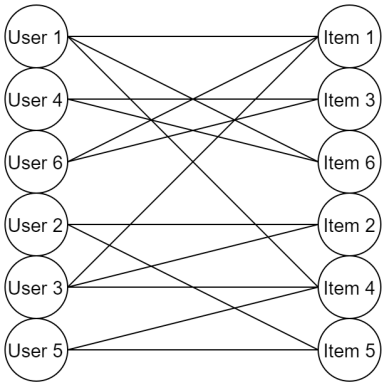
Figure 1.1: Illustration of the concepts in collaborative filtering and content-based approaches.

to recommend the unobserved items, and in order to do this, we need to predict the unobserved feedback from the observed feedback. Traditionally, two types of methods are used for this task, which are called *neighborhood-based methods* and *matrix factorization methods*.

Neighborhood-based methods can either be user-based or item-based. If they are user-based, a similarity measure, such as cosine similarity, is used to find the most similar users (called neighbors) to a target user by computing the similarity between rows in the user-item matrix. Then the top-k items for that user are determined based on the neighboring users, for example, as the items with the highest ratings or the most purchases among the neighbors. Similarly, if they are item-based, the similarity is computed between columns and the top-k items are determined based on the user's own ratings/purchases on neighboring items. Neighborhood-based collaborative filtering has the advantages of being simple to implement and the recommendations are easy to explain. The disadvantage of these methods is that they do not work very well when the user-item matrix is sparse. For example, if none of the nearest neighbors for a user have interacted with a particular item, it is not possible to estimate a score for this user-item pair, or if only a few neighbors have interacted with this item the prediction will not be robust. In other words, these methods might lack full coverage of predictions, meaning that they can not estimate scores for all

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	1			1		1
User 2		1			1	
User 3	1	1		1		
User 4			1			1
User 5				1	1	
User 6	1		1			

(a) Matrix representation.



(b) Graph representation.

Figure 1.2: Example of representations with unary feedback. For example, user 1 has given feedback to item 1, 4 and 6, but no feedback to item 2, 3 and 5.

user-item pairs.

Matrix factorization methods have a higher level of coverage even for sparse matrices. These methods use a dimensionality reduction technique, like non-negative matrix factorization, to project the user-item matrix into a lower dimensional space by exploiting the correlations between rows and columns in the original matrix. The lower dimensional representation is a decomposition of the user-item matrix into a set of latent factors representing row correlation and a set of latent factors representing column correlations. These sets of latent factors correspond to user factors and item factors, respectively, and they represent concepts in the user-item matrix, for instance, the affinity of a user towards the history and romance genres in a movie domain. The score of the ij user-item pair in the matrix can be estimated as the dot product of the ith user factor and the jth item factor. In other words, the matrix multiplication of the user factors and item factors is a fully specified matrix that approximates the original user-item matrix. It can thereby be used to fill the unobserved entries with scores to rank the top-k items for each user. Matrix factorization methods have a high coverage because the dimensionality reduction provides a dense low-dimensional representation in terms of latent factors. This means that even when two users have very few items in common, their low-dimensional latent factors can be compared.

1.3.1.4 Explanations of Recommendations

Explanations of recommendations try to address the problem of explaining why items are recommended to a user and serve several purposes [140]. They can provide transparency in the recommendation process and lead to more trust and credibility in the system. They

can also help users to make quicker and more informed decisions, thereby increasing user satisfaction. Finally, a new AI Act legislation establishes that this is compulsory.

The basic collaborative filtering and content-based recommender systems presented in Section 1.3.1.2 are typically explained by either item-based or user-based explanations [166]. Item-based explanations can be to tell the user that the recommended item shares similarities with other items the user has previously liked, for instance by highlighting the features they have in common. User-based explanations can be provided by saying that the user is similar to other users who also liked the recommended item, for instance by highlighting the similar users' ratings for this item. Item-based explanations have proved to be more intuitive for users to understand because users are familiar with the items they interacted with before, while user-based explanations have proved to be less convincing, because the target user may know nothing about other "similar" users at all [58].

1.3.1.5 Evaluation of Recommender Systems

Recommender systems can be evaluated using either an online or offline method [48]. Online methods include user studies, where users are recruited to interact with the recommender system and give their feedback about the recommendations. Online methods also include evaluation with users in a real deployment of the system. A typical metric to measure the effectiveness of a recommender system with real users is the conversion rate. The conversion rate measures the fraction of times that a user selects a recommended item. Typically, the users are randomly segmented into groups, then various recommender systems can be tested with each group of users, referred to as A/B testing. Ultimately, the most important goal is for the recommender system to do well with online users. However, online methods are often not feasible to use when comparing a lot of models and settings, because active user participation is expensive and time-consuming. In an offline method, historical data with feedback (e.g., ratings or purchases) are used. This method does not require access to real users. Once the data has been collected, it can be used to compare various recommendation models across a variety of settings.

In offline evaluation, the general experimental design with training, validation and test split of the dataset is used to not overestimate the performance of the recommender system [2]. Only the training set is used to build the training model, the validation set is used for model selection and parameter tuning, and the test set is used to evaluate the final model. If the same data set is used for training and evaluation, it will not reflect how the system will perform in a real-world setting with unseen data and the performance will likely be overestimated. Similarly, if the same data is used for parameter tuning and evaluation, the performance will be biased with knowledge from the evaluation data and thereby overestimated.

The data splits can be made with techniques commonly used in machine learning [2].

The data can for example be split with the hold-out technique, where a fraction of the feedback is hidden for testing and the remaining is used for training. This method tends to underestimate the performance since not all data is used for training. Another technique is k-fold cross-validation, where the data is split into k sets of equal size. One of the k sets is used for testing and the remaining sets are used for training. This process is repeated until all k sets have been used as a test set. The average performance over the k different test sets constitutes the evaluation. This method exploits all the available data, however it is more computationally expensive than the hold-out technique. Finally, the data can be split in a temporal manner by using the most recent feedback data as the test set. This technique mitigates the problem of user preferences changing over time.

Effectiveness is one of the most used measures in offline evaluation [48]. It is a concrete goal and easy to implement, therefore it is convenient to use for comparing various recommendation models across a variety of settings. Effectiveness can be measured in terms of the prediction effectiveness of a rating. Here standard metrics for evaluating regression models are used, such as mean absolute error or root mean squared error which are metrics to quantify the difference between predicted values and observed values. Effectiveness can also be measured in terms of the effectiveness of ranking the items, which is particularly relevant for recommender systems that output ranking of items instead of predicting ratings. In this case, it is usually each user's top-k recommended items that are evaluated. For example, the 10 items at the top of a user's ranked list, because these items will be shown to the user at the user interface. Different ranking metrics can be used to evaluate this effectiveness. For example, hit rate that measures the percentage of users in the test set where a true item was among the user's list of top-k items. Another metric that further accounts for the rank of a true item in the recommended list is mean reciprocal rank. It measures the reciprocal rank of the first true item in a user's list and then computes the mean of this reciprocal rank across all the users in the test set.

Finally, several other measures have short and long-term impacts on the conversion rate as well as the user experience [48]. One such measure is novelty which measures the likelihood of a recommender system to give recommendations that a user has not seen before. This is a distinction from items that the user was already aware of but had not chosen to interact with. Novel recommendations increase the ability of a user to discover something new. Another example is trust, which measures a user's faith in the recommendations. Even if the recommended items are accurate, they are often not useful if the user does not trust the system and thereby does not use the recommendations. Explanations of the recommendations can lead to more trust in a recommender system because they provide a logic behind the recommendations.

1.3.2 Specific Challenges for Complex Item Recommendations

In complex item domains, the characteristics of heterogeneous and multi-modal data, sparse information about the users and temporal needs play a critical role. Therefore, we dive into the theoretical background of these specific challenges in recommender systems.

1.3.2.1 Heterogeneous and Multi-modal Data

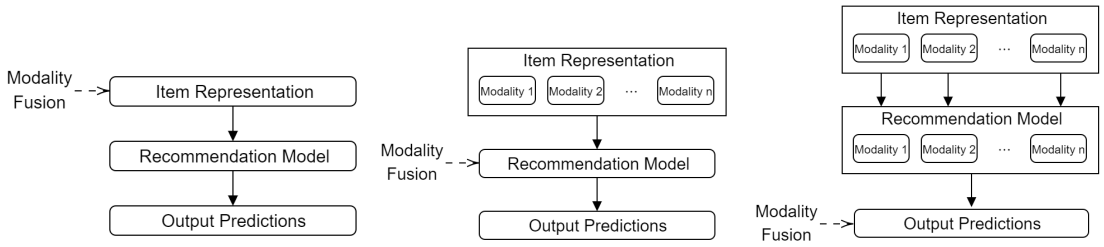
Many recommender systems use only a single type of feedback when inferring users' preferences (e.g., ratings or purchases of items), as it is usually a strong signal to represent users' preferences when there is enough of it [33]. However, users provide oftentimes more than one type of feedback on items. For example on an e-commerce platform users can click, add-to-cart and purchase items. Here purchase can be seen as the target feedback representing a strong preference for an item and the others as auxiliary feedback representing weaker preferences. This multi-typed feedback potentially provides extra information about user preferences. Therefore some collaborative filtering methods, also referred to as multi-behavior recommender systems, try to incorporate heterogeneous feedback on items [33]. There are different strategies to do so.

The most simple strategy is to encode the heterogeneous feedback as ratings, for example, click is one, add-to-cart is two and purchase is three. A similar strategy is to encode all the types of feedback as positive feedback but assign them different weights. The weights can either be manually predefined or automatically learned from data, for example based on correlations between the different types of feedback [31, 106, 113].

Some recommendation methods require both positive and negative feedback and can thereby not deal with unary feedback (i.e., the user can only specify a positive preference for an item) [64]. In this case, negative sampling can be used to randomly select some items for each user and assign them negative feedback. However, the fact that a user has not provided feedback for an item can both be because the user disliked the item or the user was not aware of the item. Therefore, some methods use information from the auxiliary feedback to better select negative samples [87, 95, 38].

As described in Section 1.3.1, the recommendation task can either be solved by estimating a score for each user-item pair or by predicting the ranking of items for a user. In the latter case, the model is trained on historical data by minimizing the difference between the predicted rankings and the true rankings. Some methods incorporate the auxiliary feedback in the true rankings, for example by ranking purchases over add-to-cart and items clicked over items with no feedback [93, 39].

Finally, transfer learning is often employed to transfer the auxiliary feedback to assist the learning task with the target feedback. For example, all types of feedback can be used collectively to generate a list of candidate items, after which only the target feedback is used to choose among the candidate items [107]. Knowledge can also be transferred with



(a) Early Fusion (b) Intermediate Fusion (c) Late Fusion

Figure 1.3: Steps of Multi-modal Recommendation Methods

multi-objective learning, where a model that generates multiple lists of items is trained simultaneously [167]. For example, lists with items that the user will click, add-to-cart and purchase

Some recommender systems also deal with multiple modalities, where a modality refers to the way in which information is expressed such as vision or speech [128]. That concerns content-based recommender systems when the items are associated with features of multiple modalities, for example when an item is presented to the users with an image and a textual description. There are some advantages of using multiple modalities [128]. Firstly, using multi-modal information could capture the same pattern from each modality resulting in a more robust model. Secondly, the different modalities can capture complementary information that is not present in a single modality. For example, an image can reflect the user's preference for the look of a jacket while a textual description can reflect the user's preference for the functionality of a jacket.

The methods for using multiple modalities in recommender systems can be categorized into late fusion [152, 25, 94], intermediate fusion [134, 163, 28] and early fusion [92, 91]. See Figure 1.3 for an illustration of the steps. Late fusion means that a recommendation model is built for each of the modalities and the model outputs are fused, for instance by averaging the predictions. With this method, each modality is independent of others. A disadvantage, however, is that it cannot learn interactions between the modalities, since the modalities are not jointly modeled in a single recommendation model. It can thereby not capture if combinations of information from different modalities lead to a user liking specific items. Intermediate fusion means that item representations are first built for each of the modalities. For example with a pre-trained text embedding model and image embedding model, respectively. Then the item representations are concatenated and fed into the recommendation model. This method might cause data redundancy, as the same information about the items can lie in multiple representations from different modalities. Finally, early fusion means that the data of different modalities are first concatenated, and then a joint item representation is built, for example with an autoencoder.

This thesis advances approaches for recommendation models when the amount of

feedback data is limited. In complex item domains, each user has a small number of historical interactions with the items, making existing collaborative filtering approaches challenging to use. On the other hand, complex items are not always associated with descriptive features that reflect the users' preferences like the color of clothes and the genre of movies, so content-based approaches (that are less dependent on feedback data) are rarely appropriate. For that reason, a general theme that runs across our contributions is to advance collaborative filtering approaches to infer recommendations from other types of user actions that are not interactions with items. This could for instance be the action of a user updating personal information or reporting a claim in the insurance domain. In *Paper 1* (Chapter 2), we present a method to weigh different types of user actions when using a graph-based collaborative filtering method. In *Paper 2 and 3* (Chapter 3 and 4), we present a way to incorporate various user actions in a session-based collaborative filtering method and map them to purchases outside sessions. In *Paper 4* (Chapter 5), we present a method that infers recommendations from user actions that not only are of different types but also different modalities, as user actions often are of different modalities like clicks and speech in complex item domains (see Section 1.1).

1.3.2.2 The Sparsity Problem

As described in Section 1.3.1, the basic approaches for recommender systems are challenged when there is not enough historical user feedback on items. This is called the *sparsity problem* in recommender systems, and it is particularly prevalent for collaborative filtering. This is because it relies on a lot of feedback from both the user of interest as well as other users to infer recommendations [68].

As also described in Section 1.3.1, collaborative filtering methods make use of the user-item matrix representation, and compute either similarities between rows/columns, or use matrix factorization to decompose the matrix into a lower dimension, and then reconstruct it as a fully specified matrix. Another way to represent the feedback is as a graph [148]. In this case, users and items are vertices in the graph and there is an undirected edge between a user and an item if the user has interacted with this item (possibly weighted by the rating if the feedback is explicit). See Figure 1.2b for an example. Then the most similar users to a target user are the ones that are encountered frequently in a random walk in the graph starting at that user. The advantage of the graph-based method is that it is more effective than the traditional collaborative filtering methods when the user-item matrix is sparse. The reason for that is that with the graph-based method, two users do not have to have liked many of the same items to obtain a high similarity as long as many short paths exist between the two users. So, this method takes also indirect connectivity between vertices into account, when computing the similarity between users, and in sparse user-item graphs, direct connectivity may not exist for many vertices. Besides being able to handle the traditional collaborative filtering problems, the graph-based methods are also

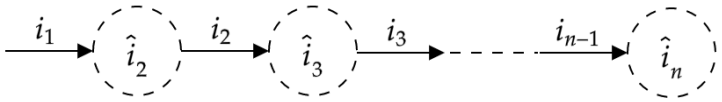


Figure 1.4: The task of a sequential or session-based recommender system is to predict the next item $\hat{i}$, the user is going to interact with for every step in a user's sequence of interactions with items $\{i_1, i_2, i_3, ...\}$

able to model various data relationships. The methods can correlate the user's preference for content, while also correlating connections between contents or users by incorporating knowledge bases or social networks in the graph.

In *Paper 1* of this thesis (Chapter 2), we present a novel graph-based recommendation model that is effective in sparse domains and improves over existing graph-based models by using a heuristic method to automatically optimize weights in the graph.

1.3.2.3 Temporal Recommendations

The basic approaches for recommender systems use all historical feedback from a user and do not take into account when the feedback was given. As a result, these models learn long-term user preferences that are static over time. However, users' preferences likely change over time and there might be information lying in the order in which a user interacts with items, for example, a user more likely purchases a laptop sleeve after the user purchased a laptop. There are different methods to generate more temporal recommendations, which are presented next.

Many forms of implicit feedback, such as buying or streaming behaviors, are inherently temporal. Therefore, *sequential recommender systems* have become increasingly popular in academic research and practical applications [147]. In this case, sequential patterns in users' historical actions are used to provide recommendations that reflect dynamic preferences.

Sometimes this implicit feedback is naturally grouped into sessions, defined as multiple user actions that happen together within a defined period of time, for example, products clicked in one transaction visit. A sub-type of sequential recommender systems is *session-based recommender systems* that base the recommendations only on the actions in a user's ongoing session, thereby reflecting the user's short-term preferences [96]. This type of recommender system is particularly relevant in many real-world applications where long-term information is not available because users are anonymous or there are many first-time users.

The recommendation task in sequential and session-based recommender systems can be formulated as the task of predicting the next item for a user for every step in the user's interaction sequence, with the output usually being a top-k ranked item list. See Figure 1.4

for an illustration. The earliest and most simple methods applied for this task include sequential pattern mining, Markov models and neighborhood-based methods [6, 56, 67]. Pattern mining strives to use association rules to mine frequent patterns with sufficient support and confidence [6]. In the sequential and session-based recommendation task, patterns refer to the sets of items that are frequently co-occurring within a sequence in a specific order. Although this method is easy to implement and relatively simple to explain to users, it is very time-consuming to match patterns in a large number of sequential combinations and the method is limited to univariate discrete sequences. In Markov models, the sequential information is encoded in the form of states [56]. A k order Markov model defines a state based on the last k items the user interacted with. The probability of transition from one state to another is estimated for all the states on historical data and is used to predict the next item for a user given the current state. In order to reduce computation time, the Markov method relies on the memoryless property. The idea is to choose a small number for k such that the recommendation depends only on the last few interactions. This constraint makes the method unable to leverage the dependencies among items in long sequences. The neighborhood-based methods are similar to the traditional neighborhood-based collaborative filtering presented in Section 1.3.1, but estimate either item neighborhoods by finding items that often occur together within sequences or sequence neighborhoods by computing the similarity across entire sequences [67]. These methods do not consider the sequential dependencies among items.

More recent methods applied to the sequential and session-based recommendation task are based on neural networks such as recurrent neural networks and attention networks [43]. These methods efficiently train a parametric model that can capture non-linear and high-order sequential dependencies between items and are flexible to deal with multivariate and continuous sequence information. However, the neural methods are known as data-hungry and non-interpretable because of their complexity. Neural networks have a huge number of parameters, and the interactions between these parameters are non-linear. This makes it prone to overfitting if it is not trained on enough data and difficult to understand how each parameter contributes to the final prediction.

In this thesis, we introduce the first-ever session-based approach for insurance recommendations, in order to learn users' dynamic needs of complex items rather than static preferences. In *Paper 2 and 3* (Chapter 3 and 4), we use the sequence of a user's sessions on the insurance website to provide temporal context when learning recommendations. In *Paper 3* (Chapter 4), we further explore the temporal dimension by using attention mechanism to pay different attention to the sessions in a user's historical sequence. In addition, we propose a new loss function for session-based recommendation models that account for the fact that users often have multiple sessions in complex item domains before they purchase.

1.3.2.4 Explanations of Temporal Recommendations

As described in Section 1.3.2.3, sequential and session-based recommender systems do not compute item or user similarities in the same way as the basic recommendation models, and explanations can thereby not be generated with the same methods. Rather explanations of these models can be provided by highlighting how the actions in a user's sequence contributed to the recommendation, for example, "because you clicked item x and y, we recommend you item z" or "many users clicked item a after item b" [30, 168, 153]. This task is similar to the general task of explaining the prediction from a machine learning model [100]. Methods for explaining machine learning models can be classified as intrinsic or post hoc. The intrinsic approach develops more interpretable models, whose logic is to a higher degree transparent, and thus can more easily provide explanations for a prediction. The post hoc approach allows the machine learning model to be a black box and develops a separate model to generate an explanation of a prediction after the prediction has been made. The methods can also be classified as model-specific or model-agnostic. Model-specific methods are limited to specific model classes. Methods that can only explain neural networks for example are model-specific. Model-agnostic methods on the other hand can be used on any machine learning model as they usually only look at feature input and output pairs and do not have access to model internals. While post hoc methods can be both model-specific and model-agnostic, intrinsically interpretable models will always be model-specific.

The most prominent type of post hoc explanation methods are feature attribution methods. They explain a model prediction by assigning attribution scores to individual input variables that reflect how the variables contributed to the prediction. Feature attribution methods can be perturbation-based, such as input occlusion [75], which estimates the contribution of input variables to a model's prediction by occluding variables from the input and measuring the corresponding changes in the model's prediction. Some occlusion methods also marginalize over input features to be explained [27, 109]. Feature attribution methods can also be gradient-based, such as vanilla gradient [127], which computes the gradient of the output with respect to the input, and extensions of that, which improves saturation and stability problems [129, 138]. The saturation problem occurs if the network to be explained changes locally only very little since it will then produce a small gradient close to zero. The stability problem occurs because gradients of complicated functions change very quickly as you change the input.

A subgroup of feature attribution methods is additive feature attribution methods which require the sum of the attribution scores to equal the prediction. These are very popular as the scores become more intuitive for humans to understand. Additive feature attribution methods can be both perturbation-based, such as Shapley additive explanations [98], and gradient-based, such as integrated gradients [138]. Moreover, they can be simplification-based, such as local interpretable model-agnostic explanations [117]. This

method trains a simple interpretable model for each instance to be explained to approximate the local decision boundary of the original model.

Paper 5 in this thesis makes contributions to the methods for generating explanations of session-based recommendations, as no such work exists and is relevant for complex item domains where session-based recommendations are well-suited and explanations of complex product recommendations are important. In this paper, we do the first-ever study of using post hoc attribution methods to highlight how the actions in a user's sessions contributed to the recommendation.

1.4 Scientific Contributions

This section provides a detailed overview of the papers included in the thesis. The papers cover three main themes of generating recommendations and explanations of those, namely the sparsity problem, temporal recommendations and heterogeneous user actions. Table 1.1 maps the sparsity, temporal and heterogeneous contributions of each paper along the recommendation and explanation axes, which are all themes arising from recommender systems in complex item domains (see Section 1.1).

Table 1.1: Overview of the main contributions of the papers in this thesis by theme, **sparsity** problem, **temporal** recommendations and **heterogeneous** user actions, and type of contribution, generating recommendations (R) and generating explanations of recommendations (E). These themes arise from complex item domains, where users have sparse interactions with items, they have temporal needs rather than preferences for complex products and they have heterogeneous behavior in terms of website actions and phone calls.

	Sparsity		**Temporal**		**Heterogeneous**	
	R	E	R	E	R	E
Paper 1 [21]	✗				✗	
Paper 2 [20]	✗		✗		✗	
Paper 3 [22]			✗			
Paper 4 [23]					✗	
Paper 5 [24]				✗		

1.4.1 Paper 1: Graph-based Recommendation for Sparse and Heterogeneous User Interactions

Recommender systems heavily rely on previous user interactions to infer recommendations (see Section 1.3.1), and a lot of research focuses on recommendation models based on complex machine learning and deep learning methods that can be very data-demanding. However, many domains have a limited availability of user interaction data, for instance, a startup business that has not collected much data yet because it is in its early stages, or a domain like insurance that is inherently data-sparse because of few different items and the fact that users rarely interact with insurance products. This paper contributes a novel recommendation model that is able to work in data-sparse domains.

General machine learning techniques to solve small data problems include the use of transfer learning, side information or knowledge graphs [84, 155, 150, 7]. More specific to the recommendation task, graph-based methods are helpful when data is sparse (see Section 1.3.2). Previous work has tried to improve the graph-based methods by adding item or user attributes to the graph or by creating predefined paths or weights from expert knowledge [85, 165, 159, 162]. We propose a new method that does not require access to other domains, as it simply uses user interaction data and it is automatically optimized without the need for expert knowledge.

We present a graph-based recommendation model, but unlike previous methods, our model represents the user interactions in a heterogeneous graph, where edges denote different types of interactions. The edges are associated with weights representing the strength of the relationship between users and content. In that way, users can be more similar if they are connected by one interaction type rather than another. We furthermore include various types of user interactions in the graph, also interactions with content that is not recommendable, to supplement the sparse data. Finally, we use genetic algorithm to automatically find the optimal weights [61].

We evaluate our model on two data-sparse use cases: (1) An educational social network dataset collected from a startup business providing a social platform for students; (2) An insurance dataset with purchases of insurance products and clicks on the insurance website. Experimental results confirm that graph-based methods (including our proposed graph-based model and the existing graph-based baselines) perform better than the traditional collaborative filtering methods when users have few interactions with items. Moreover, our proposed model outperforms the existing graph-based models with no or predefined weights. Furthermore, our model is more robust than a complex neural graph model when there are few users in the dataset.

1.4.2 Paper 2: Learning Recommendations from User Actions in the Item-poor Insurance Domain

This paper contributes a recommendations model that deals with three important characteristics of complex item domains with the insurance domain as a focal point, namely (1) Customers have **dynamic needs** of insurance products rather than static preferences as they are often related to life events like birth of child, moving house, retiring, etc.; (2) The insurance domain is **data-sparse** because there are few different items, and because insurance products typically are bought for a long-term commitment; (3) Many users browse the insurance website but prefer to finalize the purchase over the **phone**. In addition, we publish a real-world recommendation dataset from the insurance domain to facilitate future research on this topic.

To deal with these characteristics, we propose a session-based approach that uses sessions on the insurance website to give temporal context for the recommendations, since the sessions reflect the users' recent and sequential behavior (see Section 1.3.2). Session-based recommender systems usually learn from user interactions with items. However, various other actions on an insurance website can be used as signals for inferring recommendations. For example, if a user has just tried to report a claim that was not covered by the user's current insurance products. Therefore, we use various types of actions to infer recommendations, also actions that are not associated with items. Finally, we model relationships between user sessions and insurance purchases that do not take place within the sessions.

Our approach differs from the related work in the insurance domain, where state-of-the-art is to categorize users based on insurance portfolios (i.e., what insurance products the user currently has) and demographic characteristics such as age, employment and residence, then make recommendations based on these categories [161, 122, 111, 112]. It also differs from the traditional session-based task, that is to predict the next item the user is going to interact with, for every step within the ongoing session [43].

Our proposed model is a recurrent neural network with gated recurrent units, that takes as input the ordered sequence of a user's past sessions each being a set of actions, and predicts what items the user will buy after the last session. We propose different ways of passing sessions through the recurrent neural network: (1) We encode a session by aggregating over the actions with a maximum pooling operation. Then the sequence of encoded sessions is passed through the network; (2) We concatenate all sessions of a user into a single session. Then the actions are directly passed as input to the network; (3) We use an autoencoder to automatically learn encodings of sessions. The autoencoded sessions are then passed to the network.

In an experimental evaluation on a real-world dataset, our model outperforms baselines from previous research in the insurance domain showing that users' sessions are stronger signals for insurance recommendations than portfolios and demographics. Fur-

thermore, our model outperforms state-of-the-art session-based baselines showing that the traditional session-based approach (i.e., recommending the item most likely to be the next item the user interacts with on the website) is not well suited for insurance recommendations, and our recurrent neural network approach is important for modeling the relationship between actions on the website and purchases outside sessions. Further analysis reveals, that all considered action types are beneficial for the model and that the two types of information, sessions and demographics, capture different aspects of the problem, why a hybrid approach is best.

1.4.3 Paper 3: Recommending Target Actions Outside Sessions in the Data-poor Insurance Domain

This paper is an extended version of *Paper 2* (Chapter 3) and contributes two new approaches that account for temporal aspects in recommendations of complex items and additional analyses that give particular insights into our models.

Because users often have multiple sessions on the insurance website prior to their purchase, we extend the recommendation model presented in *Paper 2* (Chapter 3) with an attention mechanism that automatically learns the importance of each session in a user's history [9]. In that way, more weight can be given to more important sessions, for instance, more recent sessions, when generating recommendations.

At the time of recommending, we do not know when a user will purchase (i.e., after how many sessions). For that reason, we further propose a new loss function for session-based recommendation models that accounts for the time of purchase. Rather than estimating the likelihood of an item being purchased after the last session, we now estimate the time to the next purchase of an item after every session in a user's sequence of sessions. We then use the Weibull distribution to define our loss function as its properties are well-suited for time-to-event data. Furthermore, we use a censored version of the Weibull distribution, since a user typically has not purchased all of the items by the end of the training period [99].

Experimental results on the insurance dataset show that the attention mechanism assigns the most importance to the last session in a user's history while the rest of the sessions are assigned equal importance. Moreover, using a loss function that accounts for the time of purchase, gives more accurate recommendations earlier in the user's sequence of sessions, which is desirable in a real-life scenario where we want to make recommendations already from the user's first session.

Combining sessions with demographic features improves recommendation accuracy, motivating and investigation of potential demographic biases in recommendations of complex items. Therefore, we perform a fairness analysis of our model, by exploring the recommendation accuracy for age, gender and income of the users, which should not be

discriminated against. The analysis shows that our models are not unfair with respect to these protected characteristics of the users.

1.4.4 Paper 4: Dataset and Models for Item Recommendation Using Multi-Modal User Interactions

Complex item domains are not only diverse in the type of user feedback (e.g., purchases and claims reporting) but also the modality of user feedback is very diverse since users interact with the companies through different channels like website and call center. Although there are various examples of user interaction modalities in recommender systems such as social tagging, image posting and location sharing [40, 80, 10], previous research on combining different modalities has exclusively focused on multi-modal recommender systems for items represented by different modalities [55, 152, 134, 91]. In this paper, we contribute a recommendation dataset with multi-modal user interactions that allows progress in this research area. Furthermore, we experimentally compare several approaches for combining multi-modal user interactions in recommendation models.

The dataset that we publish is a real-world dataset from the insurance domain with (1) 115,045 web sessions on the insurance website; (2) 25,515 transcribed conversations between users and insurance agents; (3) 62,401 purchases of insurance products. The data can be used to predict insurance recommendations based on users' web sessions and conversations with the insurance company. A particular characteristic of this dataset is that incomplete modalities naturally occur since not all users interact through all the available channels. This is a problem for existing multi-modal recommendation models which only work when all modalities are available. As our second contribution, we compare several methods from the broader field of machine learning that deal with missing modalities, such as different imputation methods [26, 141, 144] and knowledge distillation [145]. These methods deal with modalities that are missing for technical reasons like problems with the measurements. In our case, however, the modalities are missing for natural reasons, for example, some users choose to purchase items through the website, while others prefer to purchase over the phone. For that reason, we propose a novel recommendation approach that specifically deals with modalities with innate incompleteness. Our approach maps the different modalities into a common feature space such that we can jointly model user interactions of multiple modalities. A missing modality will not affect the model since they can be treated equally once they are represented in the same space.

Experimental results show that the two modalities contain different information about the users that supplement each other well in the recommendation task. Furthermore, our approach of mapping the modalities into the same representation space manages to capture important interactions between the modalities. Especially, a less frequent

modality benefits from being modeled together with a more frequent modality.

1.4.5 Paper 5: Feature Attribution Explanations of Session-based Recommendations

Session-based recommender systems are useful for complex item domains as well as many other domains [96]. Advanced machine learning models constitute the majority of solutions for session-based recommender systems to capture item relationships in short sessions. However these models are not easily explainable, and explaining why an item is recommended to a user is important to increase the trust and transparency of the recommender system. Moreover, the recent AI Act legislation requires explanations of recommendations that are understandable for humans.

This paper studies the generation of explanations in session-based recommendation models. Explanations of these models belong to the kind of explanation that highlights the contribution of the features involved in modeling, for example, words or clicks (see Section 1.3.2). In the broader field of machine learning, additive feature attribution methods have come to the fore for this kind of explanation as they can work post hoc with any trained model without affecting prediction accuracy. This is opposed to intrinsic methods that work by restricting the complexity of the model [100]. However, we posit that existing additive feature attribution methods are not ideal for explaining session-based recommendation models due to two special characteristics of these models: (1) sequential dependencies, occurring when the model finds patterns in the order in which a user interacts with items; (2) repeated interactions, occurring when a user interacts with the same item multiple times in a session. These two phenomena are not accounted for in the existing methods, as they assume that interactions occur independently from each other, which is not the case with sequential dependencies. Furthermore, they fail when the features are correlated which is the case with repeated interactions.

In this paper, we unfold the reason why additive feature attribution fails to explain session-based recommendations, we empirically confirm the extent of the problem and contribute a feature attribution approach that is designed to overcome the issue with session-based recommendation models. Our method computes joint feature attributions for sets of interactions with sequential dependencies and sets of repeated interactions.

Experiments over a variety of datasets, recommendation models and explainability methods show that additive feature attribution does not reflect the attribution coming from sequential dependencies and correlated user interactions in session-based recommendation models. Furthermore, our approach of incorporating non-additive attribution scores for these sets of interactions obtains notably more faithful explanations than state-of-the-art feature attribution methods.

1.5 Summary and Future Work

The papers included in this thesis collectively contribute in advancing methods for recommender systems for complex item domains. Our investigation of these domains highlights specific challenges in existing methods for recommender systems, namely that they rely on a large volume of user feedback, they infer users' preferences rather than needs and they are designed for homogeneous user interactions with items. In particular, we contribute (1) a work on recommendations based on sparse and heterogeneous user interactions; (2) a work on inferring recommendations from web sessions with various types of user actions; (3) an extension on temporal recommendations based on web sessions; (4) a dataset and methods for learning recommendations from multi-modal user interactions; and (5) a study of explaining session-based recommendations. Overall in this thesis, we have reviewed the literature on sparse, temporal and heterogeneous recommender systems as well as explanation methods of these. Then we have evaluated existing methods under the special conditions arising from complex item domains. Finally, we have empirically validated our findings on real-world datasets that we have made freely available to the research community.

While this thesis has focused on the generation of recommendations, there are still questions left unanswered regarding the use of a recommender system in complex item domains like the insurance domain. One of the purposes of a recommender system in such domains is to help customers through the company's digital channels where there is no service agent to advise them. The advantages of having more customers buying online include the possibility to sell on a 24-hour basis, sell on bank holidays, and ease the pressure on the service agents so they have time to focus on the difficult cases only. While one important step is to learn what products a customer needs, another important step is to learn what information we should provide to the customer along with the recommendation for the customer to be able to make an informed decision. Papers 2 and 3 (Chapters 3 and 4) showed for example that 75% of the customers in the insurance domain started to browse on the insurance website but ended up purchasing over the phone. If we learn what information a customer needs, we can accommodate the recommendations on the website with that information. We can thereby help the customer to finalize the purchase on the website. An additional question is how can we personalize the way we present recommendations and information to customers on digital channels. For example, it may differ how familiar customers are with insurance products. While some customers prefer a high degree of detail, other customers find it very complex and prefer simplicity. In this case, textual descriptions of products, terms of policy, payment processes and so forth can be formulated with a different degree of complexity depending on the customer's preference.

When evaluating the generated recommendations, this thesis has focused on effec-

tiveness in terms of hit rate and other ranking evaluation metrics on historical data. A risk of focusing on effectiveness only is that we persuade the customers to buy insurance products they may not need. Another direction for future work is to explore measures for evaluating the long-term effect of recommending complex items. For instance, how it affects the cancellation rate of insurance products or the actual use of insurance products. Optimizing the recommendation model with respect to such kind of metrics could help avoid persuasion to buy products that the customers do not need.

With the rise of recommender systems, movies, books, news, and other items concerning many aspects of life are generated by an algorithm rather than a person. This has led to new challenges from an ethical perspective including threats to individual autonomy, possible negative social effects, fairness, system opacity and privacy violations.

The concern of individual autonomy and social effects arises from the influence of recommender systems on users' decision processes. Since recommender systems present only selected items to a user and keep the rest less visible to the user, the user tends to interact with the recommended items only, regardless of the items' actual relevance. This is problematic if the recommendations are inaccurate or manipulated to promote business goals as users are pushed toward items that do not reflect their interests and original intentions. Even if the recommendations are relevant, the user might miss important aspects in their decision-making by only considering the items selected by the system. Since recommender systems can have such strong influences on users, they eventually can impact society. This ethical challenge requires awareness, not least when recommender systems are deployed in complex item domains as these domains can be morally loaded. For example, consider the impact a real estate recommender system could have on the housing market in specific areas. It could potentially lead to scarcity of certain types of housing and population groups who cannot find affordable housing anymore. In relation to that, the issue of fairness concerns recommender systems that perform differently across demographic divisions constituted by characteristics such as gender and nationality. This can also affect society in terms of systematic discrimination. For example, certain users may not receive recommendations for certain housing based on wrong or biased predictions.

System opacity is the lack of transparency in the system's decision making and it is closely connected to recommendation decisions that are not explainable to humans. Opacity is an ethical issue because if the reason behind a recommendation cannot be understood it is difficult for users to object to recommendations they find wrong or unfair. While this thesis has taken a step towards better explanations of temporal recommendations, the influence of sparsity and heterogeneity on the quality and generation of explainable recommendations is still left for future work.

Finally, privacy violations arise because of the need for a recommender system to access users' private data such as user behavior. This can be a threat to user privacy because users are not always properly aware of the collection and use of this data and because

of the risk of the data being leaked to third parties. This thesis contributes methods for generating higher-quality recommendations from sparse data and without the use of demographic data. It thereby paves the way for recommender systems to collect less private data about users without compromising the quality of the recommendations.

Chapter 2

Graph-based Recommendation for Sparse and Heterogeneous User Interactions

Abstract

Recommender system research has oftentimes focused on approaches that operate on large-scale datasets containing millions of user interactions. However, many small businesses struggle to apply state-of-the-art models due to their very limited availability of data. We propose a graph-based recommender model which utilizes heterogeneous interactions between users and content of different types and is able to operate well on small-scale datasets. A genetic algorithm is used to find optimal weights that represent the strength of the relationship between users and content. Experiments on two real-world datasets (which we make available to the research community) show promising results (up to 7% improvement), in comparison with other state-of-the-art methods for low-data environments. These improvements are statistically significant and consistent across different data samples.

2.1 Introduction

With the advent of the internet, huge amounts of data have become available. This allows to design and develop novel Recommender Systems (RSs) based on complex Machine Learning (ML) and Deep Learning (DL) approaches, often characterized as data-hungry approaches. Many recent recommender models belong to this category, so a recommender dataset of size 100K might already be considered small [82]. Moreover, when using such datasets, a pre-processing step is often applied to remove all users with less than a certain

number of interactions, e.g., 5, because several models are not able to learn with only few data points per user [137, 97, 83, 52].

In the era of big data, Small and Medium Enterprises (SMEs) struggle to find their way, given that they might not have access to such a huge amount of data. However, SMEs are fundamental actors in the global economy, as they represent about 90% of businesses and more than 50% of employment worldwide [47]. In these cases, RSs able to cope with low data scenarios are necessary [51].

In ML and DL, small data problems are notoriously hard and are usually solved with a number of well-studied techniques [102]: (1) data augmentation, where synthetic samples are generated from the training set [86, 146, 158]; (2) transfer learning, where models learn from a related task and transfer the knowledge [84, 155]; (3) self supervision, where models learns from pseudo or weak labels [154, 126]; (4) few-shot learning, i.e., (meta-)learning from many related tasks with the aim of improving the performance on the problem of interest [135, 133, 123]; (5) exploiting prior knowledge manually encoded, for example external side information and Knowledge Graphs (KGs) [150, 7]. However, except for hand-coded knowledge, the above approaches still require a considerable amount of initial data or access to a different, but similar domain, where plenty of data is available. On the other side, knowledge bases are application dependent, require access to expert knowledge, and are not always available.

In this paper, we **contribute** a novel recommender approach able to operate in small data scenarios: our model does not require large volumes of initial data and is not application dependent. We use a heterogeneous graph, where vertices denote entities, e.g., users and different types of content, and edges represent interactions between users and content, e.g., a user posting a message on a social media. Then we use Personalized PageRank (PPR) to recommend items. Note that, edges represent any interaction with users and content, not only interaction with recommendable items. We assign weights to edges in the graph, which represent the strength of the relationship between users and content. In previous work within RSs [85, 159, 162], such weights are usually pre-defined depending on the application. We do not make any assumption on the values of such weights and optimize them with a genetic algorithm [61]. To the best of our knowledge, heuristic algorithms have never been applied to learn edge weights in the context of RSs. Our approach is evaluated on two real-world use cases: (1) an emergent educational social network, where there are few user interactions due to the initial stage of the platform, but a large number of items; (2) an insurance e-commerce platform, where there are many users but few user interactions, because users do not interact often with insurance products, and few items by nature of the insurance domain. Experimental results are promising, showing up to a 7% improvement over state-of-the-art baselines.

2.2 Related Work

Recommendation with small data has been tackled heuristically, i.e., by recommending items based on a set of specific rules [66]. Such rules have to be designed for each use case, making these models application dependent. Hybrid RSs have also been proposed for small data, for instance by merging Content Based (CB) and association rules [76, 77]. Note that the datasets in [76, 77] are not publicly available. Item-to-item recommendation is addressed in [125] with a CF approach as a counterfactual problem, where a small collection of explicit user preferences is used to improve propensity estimation. We cannot use this in our work because: (1) our task is not item-to-item recommendations; (2) we do not have access to explicit user preferences; (3) a large dataset (MovieLens 25M [54]) is still needed to estimate propensity (the small annotated dataset is only used to debias the propensity estimate). In [131], a hybrid user-based model combines CF, rule-based recommendation, and the top popular recommender with domain-specific and contextual information in the area of a small online educational community. The dataset is not publicly available and the approach is domain-dependent, hence not applicable to our work. Finally, conversion rate prediction for small-scale recommendation is used in [108], with an ensemble of deep neural networks that are trained and evaluated on a non-public dataset of millions of users, impressions, and clicks. Our small data scenario does not include enough data to train this ensemble model.

Solutions for cold start cases (where users or items have few or no interactions) are hybrid combinations of CF, CB, demographic and contextual information [115, 11, 50], or ML methods such as data augmentation [158], transfer learning [5, 155], etc. (see §2.1). Data augmentation is used in [86], where a CF model creates synthetic user ratings and is then combined with a CB model. We cannot use this in our task because we do not have explicit ratings (we use any user interaction as implicit feedback). In [154], self-supervision and data augmentation are combined on the user-item graph, and in [126], self-supervision on the user-item graph is enhanced with features extracted from user reviews. Few shot learning and meta-learning have also been used. In [123], a neural recommender is trained over head items with frequent interactions, and this meta-knowledge is transferred to learn prototypes for long-tail items. In [133], recommendations for cold users are generated with a meta-learner that accounts for interest drift and geographical preferences. In [7], knowledge bases (KG) are used to enrich feature representations, and in [150] a neural attention mechanism learns the high order relation in the user-item graph and the KG.

All above approaches [158, 5, 155, 154, 126, 123, 133, 7, 150] are evaluated on popular publicly available datasets, e.g., MovieLens [54], Yelp, Amazon, CiteULike [143], Weeplaces, etc. These datasets are much larger than those in our case (see Tables 2.1 and 2.2) and allow using self-supervision, few-shot learning, attention mechanisms, and other neural models that we cannot use due to the extremely low amount of data. Trans-

fer learning and domain adaptation require large amounts of training data from a similar task or a related domain, which are not (publicly) available for our use cases.

Lastly, graph-based RSs can be robust as they enable information to propagate through vertices, unlike matrix completion which is affected by data sparsity [148]. This motivates recent approaches using GNN [149, 121, 126, 154, 158]. However, these are not applicable to small data problems because there are not enough samples to train GNN models. PathRank [85] uses a heterogeneous user-item graph with additional vertices that are attributes of items, e.g., movie genre, director, etc. Recommendations are generated with a random walk similar to PPR, but constraints are used to ensure that the random walks follow certain predefined paths. These are application dependent. In [165], the user-item graph is extended with item attributes, and meta-paths are defined to determine how two entities in the graph (vertices of different types) are connected (this encodes entity similarity). A preference diffusion score is defined for specific meta-paths, based on user implicit feedback and co-occurrences of entities, and used to recommend items. Unlike [85, 165], we build the heterogeneous graph from all user interactions, not only interactions with items. We also do not include item attributes in the graph and we do not use predefined paths or meta-paths. We assume any possible path and optimize edge weights with a genetic algorithm.

Injected Preference Fusion (IPF) [159] extends PPR with a session-based temporal graph (STG) that includes both long- and short-term user preferences. STG is a bipartite graph where users, items, and sessions are vertices. Non-negative weights are associated with edges, which control the balance between long- and short-term preferences. Multi-Layer Context Graph (MLCG) [162] is a three-layer graph, where each layer represents a different type of context: (1) user context, e.g., gender and age; (2) item context, e.g., similarity between items; and (3) decision context, e.g., location and time. Different weights are associated with intra- and inter-layer edges, defined as functions of vertice co-occurrence. Unlike [159, 162], (1) we represent in the graph all user interactions (not only those with recommendable items); (2) we do not consider temporal, contextual, or demographic features, which may not be available; (3) we do not use predefined weights, but we optimize them with a genetic algorithm.

2.3 Approach

Usually, graph-based recommendation consists of 2 steps: (1) building the graph structure (§2.3.1), and (2) recommending items (§2.3.2). We introduce an intermediate step between (1) and (2), where we optimize weights associated to different edge types (§2.3.3).

2.3.1 Heterogeneous graph representation of user interactions

Let us consider a set of users who interact with content of various types, for example posts and comments in social networks or items and services in an online store. We represent users and content (vertices), and their relationships (edges) with a heterogeneous graph [136]. Vertices belong to different types, e.g., users, items, posts, etc. Edges have different types depending on the action that they represent, e.g., the edge with the type "like" can connect a user with a post. Moreover, edges have a direction because some actions are not symmetric, e.g., a user can follow another user, but not be followed by the same user. Note that all types of user interactions are included in the graph, i.e., also interactions with objects that are not recommendable items, for example, a user creating a post or reporting a claim.

A heterogeneous graph, or heterogeneous information network, is a type of directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where vertices and edges represent different types of entities and relationships among them. Each vertex $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ is associated to its type through a mapping function $\tau\colon \mathcal{V} \to \mathcal{A}$ and $\phi\colon \mathcal{E} \to \mathcal{R}$ respectively. $\mathcal{A}$ is the set of vertex types, or tags, e.g., users or various type of content. $\mathcal{R}$ is the set of edge types, e.g., a user liking a post.

We denote edges as $e = (i, j)$, where i and $j \in \mathcal{V}$ and $i \neq j$. Edge types are mapped to positive weights representing the strength of the relationship between two vertices. Formally, given an edge (i, j), we define a weight function $W\colon \mathcal{R} \to \mathbb{R}^+$ such that $W(\phi((i, j))) = w_{i,j}$, with the constraint that $w_{i,j} > 0$ (§2.3.3 explains how to compute optimal weights $w_{i,j}$).

Since G is a directed graph, $\phi((i, j)) \neq \phi((j, i))$, i.e., $\mathcal{R}$ contains distinct types for ingoing and outgoing edges. Therefore, each user interaction is represented as two weighted edges[1], $w_{i,j}$ from the user to another vertex and $w_{j,i}$ from that vertex to the user vertex, e.g., a user liking a post and a post being liked by a user. The weights for those outgoing and ingoing edges might differ. Edges can also exist between two content vertices when two entities are related (for example a comment that was created under a post). In case of multiple interactions between 2 vertices, e.g., a user can both create a post and like it, we define a different type of edge, i.e., a new value in $\mathcal{R}$, which represents the two actions. The weight of such edge corresponds to the sum of the weights of each individual type, e.g., the creation and liking of a post. In practice, this happens only for a few actions and does not affect the size of $\mathcal{R}$ significantly.

Figure 2.1 illustrates an example of a heterogeneous graph in an educational social network, where user 1 follows course 1 (note that course means university course in an educational setting), user 2 follows course 2, and user 1 and 3 follow user 2. Furthermore, user 2 has created post 1, user 1 has created comment 1 under post 1 and user 3 has liked comment 1.

[1]Except for non-symmetric actions, e.g., following a user.

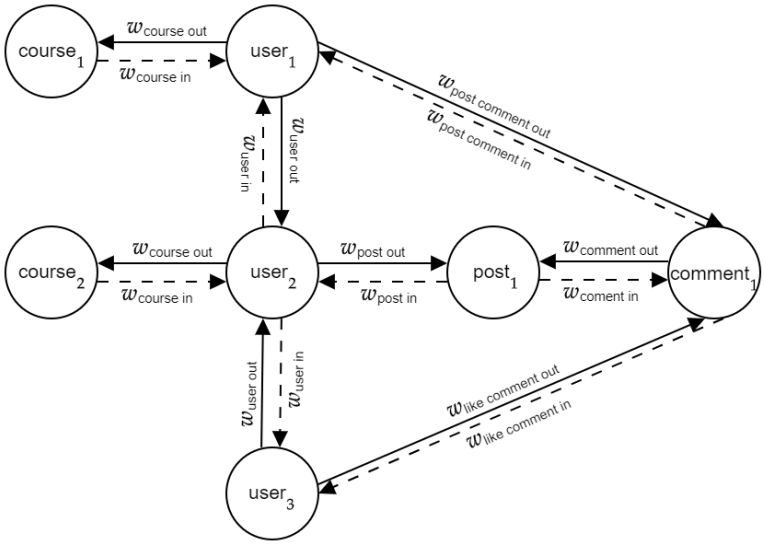


Figure 2.1: Example of the heterogeneous graph in a social network.

2.3.2 Generating recommendations using random walks

Given a user, the recommendation task consists in ranking vertices from the heterogeneous graph. To do this, we use PPR [105]. Starting at a source vertex s, the PPR value of a target vertex, t, is defined as the probability that an α-discounted random walk from vertex s terminates at t. An α-discounted random walk represents a random walker that, at each step, either terminates at the current vertex with probability α, or moves to a random out-neighbor with probability $1 - \alpha$. Formally, let G be a graph of n vertices, let $O(i)$ denote the set of end vertices of the outgoing edges of vertex i, and let the edge (i, j) be weighted by $w_{ij} > 0$. The steady-state distribution of an α-discounted random walk in G, starting from vertex s, is defined as follows:

$$\pi = (1 - \alpha)P^T\pi + \alpha e_s \text{ where } P = (p_{i,j})_{i,j \in V} = \frac{w_{i,j}}{\sum_{k \in O(i)} w_{i,k}} \cdot \mathbb{1}_{\{j \in O(i)\}} \tag{2.1}$$

$\alpha \in (0, 1)$, P is the transition matrix, e_s is a one-hot vector of length n with $e_s(s) = 1$, and $\mathbb{1}$ is the indicator function, equal to one when $j \in O(i)$. Equation (2.1) is a linear system that can be solved using the power-iteration method [105].

Solving Equation (2.1) returns a π for each user containing the PPR values (i.e., the ranks) of all the content vertices with respect to that user. A recommendation list is then generated by either ordering the content vertices by their ranks and selecting the top-k, or by selecting the most similar neighbors by their ranks, then ordering the content by the neighbors' interaction frequency with the content. We implement both methods (see

§2.4.3).

2.3.3 Optimizing edge weights using genetic algorithm

Next, we explain how to compute the weight function W, which assigns the optimal weights for outgoing and ingoing edges of each interaction type. In our data (which is presented in §2.4.1), the number of interaction types was 11 and 9, which required to optimize respectively 22 and 18 parameters (see Table 2.3). With such a large search space, using grid search and similar methods would be very inefficient. Instead, we use a heuristic algorithm to find the optimal weights. Heuristic methods can be used to solve optimization problems that are not well suited for standard optimization algorithms, for instance, if the objective function is discontinuous or non-differentiable. In particular, we use a genetic algorithm [61] as our optimization algorithm, as it is a widely known and used algorithm, which is relatively straightforward to get started with and has been shown to serve as a strong baseline in many use cases [104]. The algorithm in [61] consists of 5 components, which in our case are specified as follows: **1. Initial population:** A population consisting of a set of gene vectors is initialized. In our case, each gene vector is a vector of weights with size $|\mathcal{R}|$, and each gene is uniformly initialized from a predefined range. **2. Fitness function:** Each of the initialized gene vectors is evaluated. In our case, recommendations are generated with PPR as described in §2.3.2 where the graph is weighted by the genes. The fitness function can be any evaluation measure that evaluates the quality of the ranked list of recommendations, e.g., normalized Discounted Cumulative Gain (nDCG), Mean Average Precision (MAP), etc. **3. Selection:** Based on the fitness score, the best gene vectors are selected to be parents for the next population. **4. Crossover:** Pairs of parents are mated with a uniform crossover type, i.e., offspring vectors are created where each gene in the vector is selected uniformly at random from one of the two mating parents. **5. Mutation:** Each gene in an offspring vector has a probability of being mutated, meaning that the value is modified by a small fraction. This is to maintain diversity and prevent local optima. Finally, new offspring vectors are added to the population, and step 2 to 5 are repeated until the best solution converges.

2.4 Experiments

Next we describe the experimental evaluation: the use cases and datasets in §2.4.1; training and evaluation in §2.4.2; baselines and hyperparameters in §2.4.3; and results in §2.4.4. The code is publicly available[2].

[2] `https://github.com/simonebbruun/genetically_optimized_graph_RS`

Table 2.1: Dataset statistics: educational social network.

Type of interaction	Follow		Post		Comment		Source		Join University
	Course	User	Create	Like	Create	Like	Create	Rate	
Training	1578 (28.12%)	842 (15%)	92 (1.64%)	339 (6.04%)	116 (2.07%)	96 (1.71%)	75 (1.34%)	113 (2.01%)	1400 (24.95%)
Validation	415 (7.39%)								
Test	546 (9.73%)								

Table 2.2: Dataset statistics: insurance dataset.

Type of interaction	Purchase items	E-commerce		Personal account		Claims reporting		Information	
		Items	Services	Items	Services	Items	Services	Items	Services
Training	4853 (13.65%)	6897 (19.4%)	1775 (4.99%)	287 (0.81%)	17050 (47.96%)	154 (0.43%)	6 (0.02%)	1129 (3.18%)	2118 (5.96%)
Validation	601 (1.69%)								
Test	680 (1.91%)								

2.4.1 Use Cases and Datasets

To evaluate our model, we need a dataset that satisfies the following criteria: (1) interaction scarcity; (2) different types of actions which might not be directly associated with items. To the best of our knowledge, most publicly available datasets include only clicks and/or purchases or ratings, so they do not satisfy the second criterion. We use two real-world datasets described next. The **educational social network** dataset was collected from a social platform for students between March 17, 2020 to April 6, 2022. We make this dataset public available[3]. The users can, among others, follow courses from different universities, create and rate learning resources, and create, comment and like posts. The content vertices are: courses, universities, resources, posts, and comments and the goal is to recommend courses. The platform is maintained by a SME in the very early stage of growth and the dataset from it contains 5088 interactions, made by 878 different users with 1605 different content objects resulting in a data sparsity of 0.996. Dataset statistics are reported in Table 2.1.

The second dataset is an **insurance** dataset [20] collected from an insurance vendor between October 1, 2018 to September 30, 2020[4]. The content vertices are items and services within a specified section of the insurance website being either e-commerce, claims reporting, information or personal account. Items are insurance products (e.g., car insurance) and additional coverages of insurance products (e.g., roadside assistance). Services

[3] `https://github.com/carmignanivittorio/ai_denmark_data`

[4] `https://github.com/simonebbruun/cross-sessions_RS`

can, among others, be specification of "employment" (required if you have an accident insurance), and information about "the insurance process when moving home". User interactions are purchases and clicks on the insurance website. The goal is to recommend items. The dataset contains 432249 interactions, made by 53569 different users with 55 different item and service objects resulting in a data sparsity of 0.853. Dataset statistics are reported in Table 2.2.

2.4.2 Evaluation Procedure

We split the datasets into training and test set as follows. As test set for the educational social network dataset, we use the last course interaction (leave-one-out) for each user who has more than one course interaction. The remaining is used as training set. All interactions occurring after the left-out course interaction in the test set are removed to prevent data leakage. As test set for the insurance dataset, we use the latest 10% of purchase events (can be one or more purchases made by the same user). The remaining interactions (occurring before the purchases in the test set) are used as training set.

For each user in the test set, the RS generates a ranked list of content vertices to be recommended. For the educational social network dataset, courses that the user already follows are filtered out from the ranked list. For the insurance dataset, it is only possible for a user to buy an additional coverage if the user has the corresponding base insurance product, therefore we filter out additional coverages if this is not the case, as per [2].

As evaluation measures, we use Hit Rate (HR) and Mean Reciprocal Rank (MRR). Since in most cases, we only have one true content object for each user in the test set (leave-one-out), MRR corresponds to MAP and is somehow proportional to nDCG (they differ in the discount factor). For the educational social network, we use standard cutoffs, i.e., 5 and 10. For the insurance dataset, we use a cutoff of 3 because the total number of items is 16, therefore with higher cut-offs all measures will reach high values, which will not inform on the actual quality of the RSs.

2.4.3 Baselines, Implementation, and Hyperparameters

The focus of this work is to improve the quality of recommendations on small data problems, such as the educational social network dataset. Therefore, we consider both simple collaborative filtering baselines that are robust on small datasets as well as state-of-the-art neural baselines: *Most Popular* recommends the content with most interactions across users; *UB-KNN* is a user-based nearest neighbor model that computes similarities between users, then ranks the content by the interaction frequency of the top-k neighbors. Similarity is defined as the cosine similarity between the binarized vectors of user interactions; *SVD* is a latent factor model that factorizes the matrix of user interactions by singular value decomposition [35]; *NeuMF* factorizes the matrix of user interactions and replaces

Table 2.3: Optimized interaction weights averaged over five runs of the genetic algorithm.

(a) Educational social network dataset.

Trained for	Course follows	
Direction of edge	Out	In
User follows user	0.95	0.86
User follows course	0.73	1.88
User creates post	1.11	1.09
User creates resource	1.27	0.55
User creates comment	0.91	1.03
User likes post	1.27	0.61
User likes comment	0.42	0.99
User rates resource	0.77	0.84
Comment under post	0.91	1.17
User joins university	0.28	1.06

(b) Insurance dataset.

Trained for	Purchase items	
Direction of edge	Out	In
Purchase items	0.24	1.05
E-commerce items	0.64	1.49
E-commerce services	1.21	1.18
Personal account items	1.06	0.36
Personal account services	0.92	0.66
Claims reporting items	0.93	0.58
Claims reporting services	0.90	1.32
Information items	1.60	0.79
Information services	0.64	0.51

the user-item inner product with a neural architecture [57]; *NGCF* represents user interactions in a bipartite graph and uses a graph neural network to learn user and item embeddings [151]; *Uniform Graph* is a graph-based model that ranks the vertices using PPR [105] with all edge weights equal to 1; *User Study Graph* is the same as uniform, but the weights are based on a recent user study conducted on the same educational social network [15]. Users assigned 2 scores to each action type: the effort required to perform the action, and the value that the performed action brings to the user. We normalized the scores and used effort scores for outgoing edges and value scores for ingoing edges. The exact values are in our source code. A similar user study is not available for the insurance domain.

All implementation is in `Python 3.9`.

Hyperparameters are tuned on a validation set, created from the training set in the same way as the test set (see §2.4.2). For the educational social network, optimal hyperparameters are the following: damping factor $\alpha = 0.3$; PPR best predictions are obtained by ranking vertices; 30 latent factors for SVD; and number of neighbors $k = 60$ for UB-KNN. Optimal hyperparametrs in the insurance dataset are: damping factor $\alpha = 0.4$; PPR best predictions are obtained by ranking user vertices and select the closest 90 users; 10 latent factors for SVD; and number of neighbors $k = 80$ for UB-KNN.

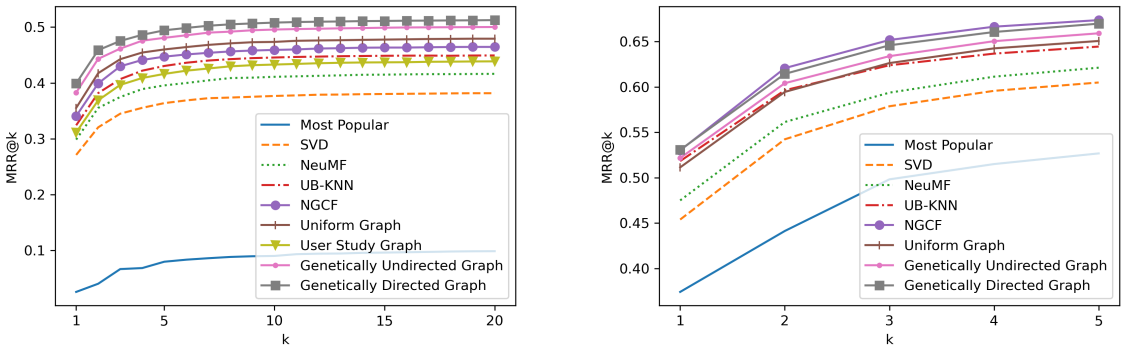
The genetic algorithm is implemented with `PyGAD 2.16.3` with MRR as fitness function and the following parameters: initial population: 10; gene range: $[0.01, 2]$, parents mating: 4; genes to mutate: 10%; mutation range: $[-0.3, 0.3]$. We optimize the edge weights using the training set to build the graph and the validation set to evaluate the fitness function. The optimal weights are reported in Table 2.3. In order to provide stability of the optimal weights, we report the average weights obtained by five runs of the genetic algorithm.

Table 2.4: Performance results (†mean/std). All results marked with * are significantly different with a confidence level of 0.05 from the genetically optimized graph (ANOVA [81] is used for MRR@k and McNemar's test [37] is used for HR@k). The best results are in bold.

Dataset	Educational social network				Insurance	
Measure	MRR@5	MRR@10	HR@5	HR@10	MRR@3	HR@3
Most Popular	0.0797*	0.0901*	0.1978*	0.2729*	0.4982*	0.6791*
SVD	0.3639*	0.3767*	0.5275*	0.6209*	0.5787*	0.7399*
NeuMF	0.3956*	0.4110*	0.5604*	0.6740*	0.5937*	0.7448*
UB-KNN	0.4304*	0.4456*	0.6172*	0.7271*	0.6238*	0.7569*
NGCF	0.4471	0.4592	0.6245*	0.7143*	**0.6517**	**0.8043***
Uniform Graph	0.4600	0.4735	0.6300*	0.7289*	0.6263*	0.7730*
User Study Graph	0.4162*	0.4330*	0.5952*	0.7179*	-	-
Genetically Undirected Graph	0.4809	0.4957	0.6410	0.7509	0.6339	0.7760*
Genetically Directed Graph†	**0.4907**/ 0.0039	**0.5045**/ 0.0037	**0.6505**/ 0.0052	**0.7520**/ 0.0055	0.6435/ 0.0029	0.7875/ 0.0044

2.4.4 Results

Table 2.4 reports experimental results. On both datasets, UB-KNN outperforms SVD and NeuMF, and the best-performing baseline is the uniform graph-based model on the educational social network dataset and the NGCF model on the insurance dataset. This corroborates previous findings, showing that graph-based RSs are more robust than matrix factorization when data is sparse [148] and neural models need a considerable amount of data to perform well. Graph-based methods account for indirect connectivity among content vertices and users, thus outperforming also UB-KNN, which defines similar users on subsets of commonly interacted items. Our genetically optimized graph-based model outperforms all baseline models on the educational social network dataset and obtains competing results with the NGCF model on the insurance dataset, showing that the genetic algorithm can successfully find the best weights, which results in improved effectiveness. In order to account for randomness of the genetic algorithm, we run the optimization of weights five times and report the mean and standard deviation of the results in Table 2.4. The standard deviation is lowest on the insurance dataset, but even on the very small educational dataset, the standard deviation is relatively low, so for different initializations, the algorithm tends to converge toward similar results. Moreover, we tried a version of our model where we let the graph be an undirected graph, meaning that for each edge type the weight for the ingoing and the outgoing edge is the same. The results show that the directed graph outperforms its undirected version. For the educational social network, weights based on the user study result in worse performance, even lower than UB-KNN. Overall, scores are higher on the insurance data. This might happen because: (1) data from the educational social network is sparser than insurance



(a) Educational social network dataset.

(b) Insurance dataset.

Figure 2.2: MRR@k for varying choices of the cutoff threshold k.

data (see §2.4.1); (2) the insurance data has a considerably larger training set; (3) there are fewer items to recommend in the insurance domain (16 vs. 388).

Figure 2.2 shows MRR at varying cutoffs k. We have similar results for HR, which are omitted due to space limitations. It appears that the results are consistent for varying thresholds. Only on the insurance dataset, we see that the UB-KNN is slightly better than the uniform graph-based model for smaller thresholds.

Inspecting the optimal weights in Table 2.3, we see that for the educational social network all the interaction types associated with courses (following course, creating resource, creating comment, creating and liking posts) are highly weighted. This is reasonable since courses are the recommended items. Moreover, a higher weight is assigned when a user follows a user compared to when a user is followed by a user. This reasonably suggests that users are interested in courses attended by the users they follow, rather than the courses of their followers. For the insurance dataset, we observe that the greatest weights are given when a user clicks on items in the information and personal account section, when items are purchased by a user, and when items and services are clicked in the e-commerce section, which are all closely related to the process of purchasing items.

We further evaluate the performance of our models, when trained on smaller samples of the insurance dataset. We randomly sample 10%, 25% and 50% from the training data, which is then split into training and validation set as described in §2.4.3. In order to account for randomness of the genetic algorithm, we sample 5 times for each sample size and report the mean and standard deviation of the results. We evaluate the models on the original test set for comparable results. The results are presented in Table 2.5. While the genetically optimized graph-based model only partially outperforms the NGCF model on large sample sizes (50% and 100%) it outperforms all the baselines on small sample sizes (10% and 25%). This shows that our genetically optimized model is more

Table 2.5: Results on smaller samples of the insurance dataset. The notation is as in Table 2.4.

Percentage of insurance dataset	10%		25%		50%		100%	
Measure	MRR@3	HR@3	MRR@3	HR@3	MRR@3	HR@3	MRR@3	HR@3
Most popular	0.5003*	0.6856*	0.4981*	0.6789*	0.5003*	0.6856*	0.4982*	0.6791*
SVD	0.5785*	0.7336*	0.5794*	0.7395*	0.5758*	0.7379*	0.5787*	0.7399*
NeuMF	0.5792*	0.7326*	0.5759*	0.7291*	0.5849*	0.7466*	0.5937	0.7448
UB-KNN	0.6183	0.7661*	0.6180*	0.7494*	0.6243	0.7524*	0.6238*	0.7569*
NGCF	0.5937*	0.7199*	0.6030*	0.7405*	**0.6397**	0.7894	**0.6517**	**0.8043***
Uniform Graph	0.6196	0.7687*	0.6206*	0.7687*	0.6253	0.7746	0.6263*	0.7730*
Genetically Undirected Graph	0.6238	0.7672*	0.6224	0.7601*	0.6286	0.7741	0.6339	0.7760
Genetically Directed Graph†	**0.6267**/ 0.0052	**0.7784**/ 0.0084	**0.6353**/ 0.0039	**0.7845**/ 0.0073	**0.6397**/ 0.0045	**0.7903**/ 0.0071	0.6435/ 0.0029	0.7875/ 0.0044

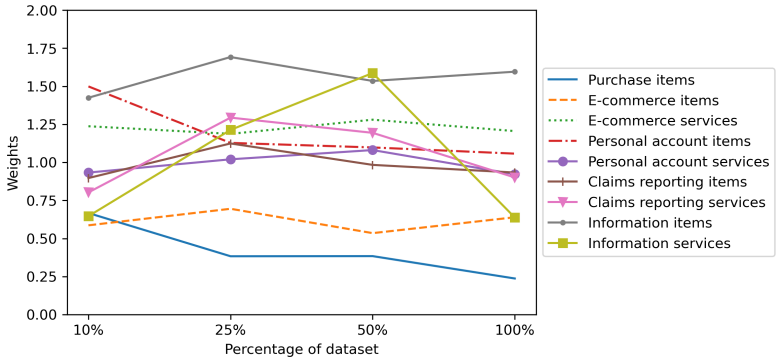


Figure 2.3: Plot of how the outgoing edge weights evolve for different sizes of the insurance dataset.

robust for small data problems than a neural graph-based model. In addition, it is still able to compete with the neural graph-based model when larger datasets are available.

In Figure 2.3 we inspect how the outgoing edge weights evolve when optimized on different sizes of the insurance dataset. We have similar results for the ingoing edge weights, which are omitted due to space limitation. It appears that the optimized weights only change a little when more data is added to the training set, and the relative importance of the interaction types remains stable across the different sizes of the dataset. Only the interaction type "information services" has large variations across the different dataset sizes, and in general, the biggest development of the weights happens when the dataset is increased from 10% to 25%. It shows that once the genetic algorithm has found the optimal weights in offline mode, the weights can be held fixed while the RS is deployed online, and the weights only need to be retrained (offline) once in a while, reducing the need for a fast optimization algorithm.

2.5 Conclusions and Future Work

We have introduced a novel recommender approach able to cope with very low data scenarios. This is a highly relevant problem for SMEs that might not have access to large amounts of data. We use a heterogeneous graph with users, content and their interactions to generate recommendations. We assign different weights to edges depending on the interaction type and use a genetic algorithm to find the optimal weights. Experimental results on two different use cases show that our model outperforms state-of-the-art baselines for two real-world small data scenarios. We make our code and datasets publicly available.

As future work we will consider possible extensions of the graph structure, for example, we can include contextual and demographic information as additional layers, similarly to what is done in [162]. Moreover, we can account for the temporal dimension, by encoding the recency of the actions in the edge weight, as done in [159]. We will further experiment with the more recent particle swarm [79] and ant colony optimization algorithms [41] instead of the genetic algorithm to find the optimal weights. Finally, we will investigate how to incorporate the edge weights into an explainability model, so that we can provide explanations to end users in principled ways as done in [8].

Acknowledgements

This paper was partially supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 893667 and the Industriens Fond, AI Denmark project.

Chapter 3

Learning Recommendations from User Actions in the Item-poor Insurance Domain

Abstract

While personalised recommendations are successful in domains like retail, where large volumes of user feedback on items are available, the generation of automatic recommendations in data-sparse domains, like insurance purchasing, is an open problem. The insurance domain is notoriously data-sparse because the number of products is typically low (compared to retail) and they are usually purchased to last for a long time. Also, many users still prefer the telephone over the web for purchasing products, reducing the amount of web-logged user interactions. To address this, we present a recurrent neural network recommendation model that uses past user sessions as signals for learning recommendations. Learning from past user sessions allows dealing with the data scarcity of the insurance domain. Specifically, our model learns from several types of user actions that are not always associated with items, and unlike all prior session-based recommendation models, it models relationships between input sessions and a target action (purchasing insurance) that does not take place within the input sessions. Evaluation on a real-world dataset from the insurance domain (ca. 44K users, 16 items, 54K purchases, and 117K sessions) against several state-of-the-art baselines shows that our model outperforms the baselines notably. Ablation analysis shows that this is mainly due to the learning of dependencies across sessions in our model. We contribute the first ever session-based model for insurance recommendation, and make available our dataset to the research community.

3.1 Introduction

Within the domain dealing with insurances for individuals such as home insurance, car insurance and accident insurance, personalised recommendations can help customers continuously adjust their insurances to suit their needs.

Recommender systems (RSs) most often learn from user feedback on items such as books purchased, movies watched, and/or ratings given to those items. These systems need a considerable amount of previous feedback on items to give high quality recommendations. This is a problem in the insurance domain, where typically there are few different items to purchase, and insurance products typically are bought to be used for a long time, so the purchase frequency is low.

Another problem in insurance recommendation is that several state-of-the-art RSs, e.g., collaborative filtering and content-based, assume that user preferences are static. However, user needs in insurance evolve over time as they are closely connected to life events like marriage, birth of child, buying a home, changing jobs, retirement, etc.

To deal with the above idiosyncrasies of the insurance domain, we focus on a session-based approach to model insurance recommendations. Session-based RSs do not use information about long-term user preferences. Instead recommendations are based on short-term user-item interactions within the ongoing session. In that way, session-based RSs capture dynamic user preferences using the session as temporal context for the recommendation. However, current session-based RSs cannot be applied to our insurance domain because many users buy insurance products over the phone *after having sessions* on the insurance website. Thus, when using past user sessions on the insurance website as signals for learning recommendations, the target action (i.e., the user's purchase of items) does not happen within the input sessions. This is an important difference from the usual session-based task, where the target action is the next item the user interacts with *in the ongoing session*.

Session-based RSs usually learn from user interactions with items e.g., view of videos or books added to cart. However, on an insurance website there are several other user actions that can be useful signals for learning recommendations. For instance, if a user has recently tried to report a claim that the user's current insurance products did not cover or if a user has recently updated his/her employment at the personal account. We thus explore the usefulness of such signals by learning recommendations from several types of user actions that are not always directly associated with items.

We present a recurrent neural network (RNN) recommendation model that learns from several types of user actions, and unlike all prior session-based recommendation models, it models relationships between input sessions and a target action that does not take place within the input sessions. We call this approach a *cross-sessions* recommendation model. Evaluation on a real-world dataset from the insurance domain and against several state-of-the-art baselines shows that our model outperforms the baselines notably. We **contribute** the first ever cross-sessions model for insurance recommendation and make our dataset publicly available.

3.2 Related Work

We present related work focusing on RSs within the insurance domain (Section 3.2.1) and session-based RSs (Section 3.2.2).

3.2.1 Insurance Domain

Previous research on RSs for the insurance domain is limited. In principle, a knowledge-based RS could be used for this task, which would mine highly personalised user information through user interactions [65], however to our knowledge no prior work reports this. Most of it supplements the small volume of user feedback with user demographics, such as age, marital status, income level, and many more. We overview these next.

Xu et al. [161] cluster users into different groups based on their demographics. Then they make association rule analysis within each group on the users' set of purchased items. Recommendations are extracted directly from the association rules. Sanghamitra Mitra [122] estimate similarities between users based on demographic attributes using a similarity measure (e.g., cosine similarity). Then they make recommendations to a user based on the feedback on items by the top-N similar users. Qazi et al. [111, 112] train a Bayesian network with user demographics and previously purchased items as input features, aiming to predict the last purchased item of a user. They further train a feed forward neural network to provide recommendations to potential users where only external marketing data is available. All these methods show more effective recommendations compared to standard RS approaches, such as matrix factorization and association rule mining solely applied to the feedback data. In addition, these methods solve the problem of cold start users occurring when recommending items to users with no previous feedback on items. Unlike these methods above, which assume that the preference dynamics are homogeneous within demographic segments, our model allows the changes in user preferences to be individual by using sessions generated by the individual user.

Bi et al. [14] propose a cross-domain RS for the insurance domain. They use knowledge from an e-commerce source domain with daily necessities (clothes, skincare products, fruits, electronics products, etc.) to learn better recommendations in the insurance target domain when data is sparse. They employ a Gated Recurrent Unit (GRU) [34] to model sequential dependencies in the source domain. Our model differs from this model by using user sessions on the website in the target domain, thereby not having the need of overlapping users across multiple domains. Moreover, the model in [14] is not session-based, but it is based on users' long-term preferences in both source and target domain.

3.2.2 Session-based Recommender Systems

Session-based RSs model users' short-term preferences within the ongoing session [43], commonly using item-to-item recommendations [36, 90]: similarities between items are computed based on the session data, i.e., items that are often interacted with together in sessions achieve a high similarity. The item similarities are then used during the session to recommend the most similar

items to the item that the user has currently interacted with. This approach only considers the last interaction of the user, ignoring even the information of the past interactions in the ongoing session. Moreover, this approach requires the target action to happen within sessions, thus this approach cannot be applied to our task.

An extension of the item-to-item approach is Session-based K-Nearest Neighbors (SKNN) [67, 62], which considers all user interactions in the session. In this approach similarities between entire sessions are computed using a similarity measure (e.g., cosine similarity). The recommendations are then based on selecting items that appeared in the most similar past session. This approach does not take into account the order of the input sequence.

A sequential extension to the SKNN method is Vector Multiplication SKNN [96], which puts more weight on the most recent user interactions of a session when computing the similarities. Another extension is Sequence and Time Aware Neighborhood [45], which uses the position of an item in the current session, the recency of a past session with respect to the current session, and the position of a recommendable item in a neighbouring session. The latter model depends on the target actions occurring within sessions, and so does not fit our task where the target actions occur outside the session.

Neural session-based approaches have also been proposed. One of the most cited is GRU4REC [59, 60]. This approach models user sessions with the help of GRU in order to predict the probability of the subsequent interaction given the sequence of previous interactions. Neural Attentive Session-based Recommendation [88] extends GRU4REC by using an attention mechanism to capture the user's main purpose in the current session. Moreover, Graph Neural Networks have been used [156]. This approach models session sequences as graph structured data, thereby being capable of capturing transitions of items and generating item embedding vectors correspondingly.

The above methods use only the user's single ongoing session. Attempts to use past user sessions when predicting the next interaction for the current session have also been proposed [114, 120, 164, 63, 110]. However, an in-depth empirical investigation of these methods [83] shows that they did not improve over heuristic extensions of existing session-based algorithms that e.g., extend the current session with previous sessions or boost the scores of items previously interacted with. Unlike all above methods, we use different types of actions, not only with items (see Section 3.4.1).

3.3 Approach

We present the problem formulation (Section 3.3.1) and our model for addressing it (Section 3.3.2).

3.3.1 Problem Formalization

The goal of our cross-sessions RS is to recommend the next items that a user will buy, given the user's past sessions. As opposed to standard session-based RSs, which predict the next step in

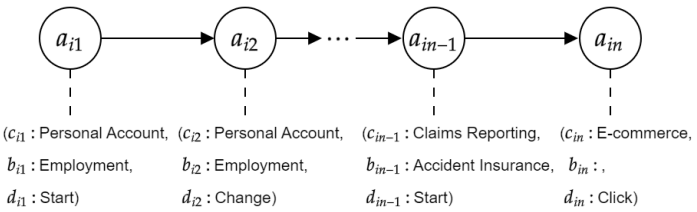


Figure 3.1: Example of session on insurance website. A session is a list of 3-tuple actions in the order of time.

the input sequence given the sequence so far, in our task: (1) the target action, i.e., purchase, occurs outside the session; (2) we use user actions across multiple sessions; (3) we use all user actions, not only actions with items. We extend the notation in [43] to accommodate these differences.

A session, s_i, is a sequence of user actions, $\{a_{i1}, a_{i2}, a_{i3}, ..., a_{in}\}$, on the website. An action, a_{ij}, is represented by a 3-tuple, i.e., $a_{ij} = (c_{ij}, b_{ij}, d_{ij})$, where:

- c_{ij}: action section, refers to the section of the website in which the user interacts;
- b_{ij}: action object, refers to the object on the website that a user chooses to interact with; and
- d_{ij}: action type, refers to the way that a user interacts with objects.

Fig. 3.1 illustrates a session where the user starts (action type d_{i1}) by interacting with employment (object b_{i1}) in the personal account section of the website (section c_{i1}). Then, the user changes (d_{i2}) the employment (b_{i2}) in the same personal account section (c_{i1}). Section 3.4.1 and Tab. 3.2 present different sections, objects, and action types.

The past sessions of a user is a list of sessions, s_i, chronologically ordered. We do not include all historical sessions of a user as we assume that only recent sessions are relevant for the current task. We use an inactivity threshold, t, to define recent sessions, and reason that two sessions belong to the same task if there is no longer than t (time) between them. We describe how we estimate the threshold t from real data in Section 3.4.1. The task is to learn a function, f, for predicting the probability that a user will buy each item k after the last session s_m based on the input sequence of user's past sessions:

$$f(s_1, s_2, s_3, ..., s_m) = (\hat{p}_1, \hat{p}_2, \hat{p}_3, ..., \hat{p}_K), \tag{3.1}$$

where each element in $\{s_1, s_2, s_3, ..., s_m\}$ is a sequence of actions, $\hat{p}_k$ is the estimated probability that item k will be bought by the user, and K is the total number of items.

3.3.2 Proposed Approach

Our model is inspired by GRU4REC [59], an RNN with a single GRU [34] layer that models user interactions with items in a single session. The RNN has as input the ordered sequence of

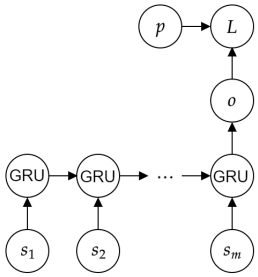


Figure 3.2: Architecture

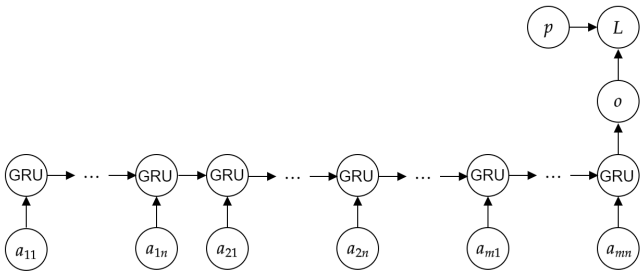


Figure 3.3: Architecture of cross-sessions concat of cross-sessions encode

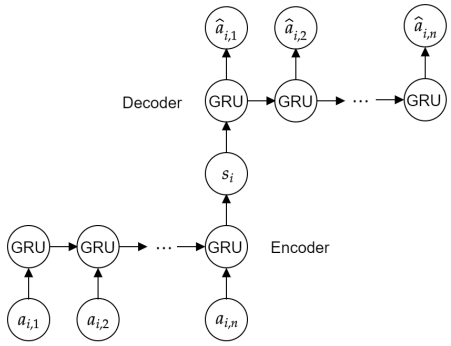


Figure 3.4: Architecture of cross-sessions auto

items interacted with in the session, and outputs for every time step the likelihood of each item being the item that the user interacts with next. Our cross-sessions RS extends GRU4REC by (1) taking multiple sessions of each user as input, as in Eq. (3.1); (2) using various types of input actions that are not always associated with items; (3) predicting what items the user will buy after the last time step as opposed to predicting the next interaction for every time step in the sequence. Next we explain different ways of passing input sessions through the RNN.

In the first way, which we call *Cross-sessions Encode* (see Fig. 3.2), we encode a session by aggregating over the actions in the session with a max pooling operation:

$$s_i = \max_{element}(a_{i1}, a_{i2}, a_{i3}, ..., a_{in}), \tag{3.2}$$

where a_{ij} is the binarized vector marking the presence of an action section, action object and action type performed by a user at time step j in session i, and $\max_{element}(\cdot)$ is a function that takes the element-wise maximum of vectors. Then, for every time step i in the sequence of a

user's sessions, an RNN with a single GRU layer computes the hidden state

$$\begin{aligned} h_i &= (1 - z_i) \cdot h_{i-1} + z_i \cdot \hat{h}_i && \text{for } \; i = 1, .., m, \\ z_i &= \sigma(W_z s_i + U_z h_{i-1}), && (\textit{update gate}) \\ \hat{h}_i &= \tanh(W s_i + U(r_i \cdot h_{i-1})), && (\textit{candidate gate}) \\ r_i &= \sigma(W_r s_i + U_r h_{i-1}), && (\textit{reset gate}) \end{aligned} \tag{3.3}$$

where W_z, U_z, W, U, W_r and U_r are weight matrices and $\sigma(\cdot)$ is the sigmoid function. The reset gate plays the role of forgetting information about the past that is not important given the current session. The update gate plays the role of judging whether the current session contains relevant information that should be stored. The hidden state, h_i, is a linear interpolation between the previous hidden state and the candidate gate.

Our second way of passing input sessions through the RNN is called *Cross-sessions Concat* (see Fig. 3.3). Here, we concatenate all sessions of a user into a single session $s = \{a_{11}, .., a_{1n}, a_{21}, .., a_{2n}, .., a_{m1}, .., a_{mn}\}$. Now the hidden state in Eq. (3.3) is computed for every time step (ij) in s. Thereby this approach takes into account the overall order of actions across sessions, whereas the cross-sessions encode only accounts for the order of sessions.

In both cases, the RNN returns an output vector, o, of length K after the last time step. Because a user can buy multiple items at the same time, we consider the learning task as multi-label classification and use the sigmoid function, $\sigma(\cdot)$, on each element of o as output activation function to compute the likelihood of purchase

$$\hat{p}_k = \sigma(o_k), \quad \text{for } \; k = 1, \ldots, K. \tag{3.4}$$

During training the loss function is computed by comparing $\hat{p}$ with the binarized vector of the items purchased, p. Due to the learning task being multi-label classification, we define the loss function as the sum of the binary cross entropy loss over all items. The loss function is thereby different from the ranking loss used in GRU4REC and is given by

$$L = -\sum_{k=1}^{K} \Big(p_k \cdot \log(\hat{p}_k) + (1 - p_k) \cdot \log(1 - \hat{p}_k) \Big). \tag{3.5}$$

We also use another variation of session encoding, *Cross-sessions Auto* (see Fig. 3.4), which automatically learns encodings of sessions with an autoencoder, instead of Eq. (3.2). We train an RNN-based autoencoder with a single GRU layer that takes as input the ordered sequence of actions in a session and is evaluated on recreating the input using categorical cross-entropy loss on each of the 3 features: action section, action object and action type. Once trained, the encoder is used to encode a session into a single vector, s_i, that can be used as input for Eq. (3.3). The architecture of the autoencoder in Fig. 3.4 is then combined with the architecture for the recommendation part in Fig. 3.2.

Finally, we use a hybrid of a cross-sessions and a demographic model where the hidden state from a cross-sessions RNN is merged with the hidden state (a dense layer) from a feed forward neural network with demographic input features of the user. The concatenation of the two is passed through a dense layer. The architecture of Cross-sessions Encode combined with demographics is illustrated in Fig. 3.5. The loss functions is cross entropy as in Eq. (3.5).

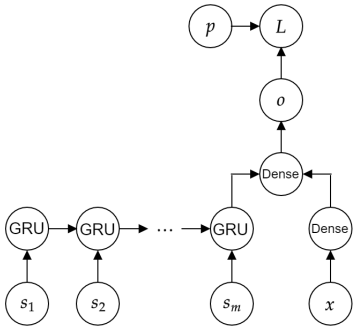


Figure 3.5: Architecture of a hybrid model between a cross-sessions and a demographic model. x denotes an input vector representing demographic features of the user.

Table 3.1: Main properties of the dataset (*mean/std).

Users	44,434
Items	16
Purchase events	53,757
Sessions	117,163
Actions	1,256,156
Purchase events per user*	1.21/0.51
Sessions before purchase event*	2.18/1.68
Actions per session*	10.72/7.85

Table 3.2: Amount of different actions in the dataset.

Action section	E-commerce	256,319 (20.41%)
	Claims reporting	11,188 (0.89%)
	Information	198,147 (15.77%)
	Personal account	790,502 (62.93%)
Action object	Items	249,378 (19.85%)
	Services	655,574 (52.19%)
	No object	351,204 (27.96%)
Action type	Click	811,747 (64.62%)
	Start	388,215 (30.90%)
	Act	47,713 (3.80%)
	Complete	8,481 (0.68%)

3.4 Dataset

To the best of our knowledge, there exists no publicly available dataset that satisfies the criteria of our setup, namely: (1) item scarcity, (2) target action happening outside the session, and (3) actions of several types that are not always associated with items. We use a real-life dataset that we have obtained from a commercial insurance vendor and that we make freely available[1] to the research community. Next, we describe this dataset.

3.4.1 Dataset Description

Tab. 3.1 shows general statistics of the dataset that was collected from the website of an insurance vendor between October 1, 2018 to September 30, 2020. During this time, there was no RS running on the website that could influence the behaviour of users. We collected *purchase*

[1] `https://github.com/simonebbruun/cross-sessions_RS`

events of insurance products and additional coverages made by existing customers. Customers were identified through log-in and cookies. Both purchases made on the website and over the phone are included. 75% of the users navigate through the company website, but still prefer to make the purchase over the phone. This resulted in 53K purchases corresponding to 44K users. We observe no major seasonal effects in the purchase frequency.

For each user in the dataset, we collected all *sessions* that occurred before the user's purchase event. A session consists of user actions on the website. The actions come with timestamps, action section (e-commerce, claims reporting, information, and personal account), action object (items and services), and action type (click, start, act, and complete). Tab. 3.2 shows all the actions and their frequency in the dataset. Most actions occur on the personal account page (63%) and the e-commerce section (20%) because the sessions are generated by existing customers in the period before they make a new purchase. Users interact mainly with services (52%). Example of services are specification of "employment" (required if you have an accident insurance), specification of "annual mileage" (required if you have a car insurance), and information about "the insurance process when moving home". Users can also click on the different sections without interacting with any objects, we denote this with "no object". Not surprisingly, most of the actions are clicks (65%) or start (31%). Examples of type "act" are "change", e.g., employment or "fill out" claims report. The action type "complete" can occur when a user completes a change, of e.g., employment, or completes a filled out claims report.

The dataset includes also *demographic attributes* and *portfolios* of each user. Demographic attributes include age, employment, income, residence, marital status, children, etc. These features are aggregated at the address level, i.e., they represent the average across users living in the same area. Portfolios represent the user history within the insurance company, e.g., items that the user has already bought. Note that our cross-sessions models do not exploit demographic attributes or portfolios, but these are needed by state-of-the-art RSs we consider as baselines (see Section 3.5.1).

This dataset is different from publicly available datasets for session-based recommender tasks, e.g., Last.fm, RecSys Challenge (RSC) datasets, etc. in the following ways. There is scarcity both in terms of items and user interactions. The total number of items is 16, much smaller than usual item sets, for example RSC15 contains 29K items and Last.fm 91K items. In terms of user interactions, there are only 1.2 purchase and 2.2 sessions per users over a period of time spanning 2 years. Motivated by the scarcity of user interactions, we decided to log all types of user actions. Therefore, we do not include only clicks and purchases/views, as in most of the available datasets, but we also record the whole user experience through the company website.

3.4.2 Estimation of Session Threshold

A common challenge when mining log data is how to determine the session length [71, 72]: we can record the start timestamp of a user session but we cannot always record the end timestamp, for example when a user leaves the browser window open without logging out. We face a similar problem but with respect to the length of the list of sessions instead of the length of a single

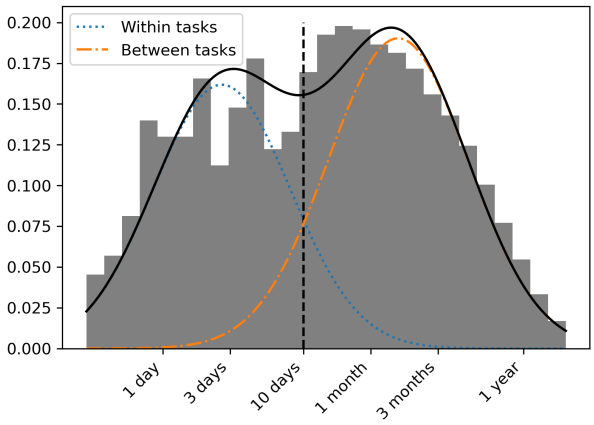


Figure 3.6: Histogram of logarithmically scaled inter-session times and fitted Gaussian mixture model.

session, i.e., we need to determine the amount of time such that two sessions can be grouped under the same task. Specifically, we need to estimate the threshold t, such that given two subsequent sessions s_1 and s_2 from the same user, if $\texttt{start\_time}(s_2) - \texttt{start\_time}(s_1) \leq t$, then s_1 and s_2 belong to the same task. This threshold will determine the maximum number of sessions to be considered by our RSs.

A rule of thumb for Web sessions is to set the session end after 30 minutes of inactivity, but a similar estimate does not exist with respect to inter sessions inactivity time. Halfaker et al. [49] estimate when a session ends directly from log data, i.e., as the amount of inactivity time that allows to deem a session as concluded. We follow a similar approach applied on the whole list of sessions instead of actions within a single session to estimate when a task ends.

First, we compute inter-session times, that is the times between consecutive sessions by the same user. Then we plot the logarithmically scaled histogram of inter sessions times (see Fig. 3.6). As reported in [49], we observe a bimodal distribution with a valley. Therefore we fit a two components Gaussian mixture model using Expectation Maximization and assume that the inter-session times are a mixture of times between: (1) sessions within the same task (blue line) and, (2) sessions belonging to different tasks (orange line). We set the inactivity threshold t equal to the point where an inter-session time is equally likely to belong to one of the two distributions. As illustrated in Fig. 3.6, the resulting threshold is 10 days. Thus two sessions belong to the same task if there is no longer than 10 days between them.

3.4.3 Dataset Pre-processing

We pre-process the data as described next. All statistics in Tab. 3.1 and 3.2 refer to the pre-processed dataset.

From the purchase events we remove items that have frequency $< 0.1\%$ and from the actions we remove sections, objects and types that have frequency $< 0.1\%$, since low frequency items

and actions are not optimal for modeling. Consecutive repeated actions of the same kind are discarded, because they very likely represent noise, e.g., a user clicking twice due to latency from the website. All sessions with < 3 actions are removed, as they are poorly informative sessions. All sessions are truncated in the end to have maximum 30 actions (the 95th percentile), to avoid very long training times. All users are kept, even those that have only a single recent session. Lastly, the 10 days threshold estimated in Section 3.4.2 motivates two additional pre-processing steps. First, sessions that exceed the 10 days threshold are discarded. Second, within the 10 days rule, the list of recent sessions for each user is truncated to maximum 7 sessions (the 95th percentile).

3.5 Experiments

We present the experimental set-up (Section 3.5.1) and the results (Section 3.5.2). Our source code is publicly available[2].

3.5.1 Experimental Set-up

First we describe the evaluation procedure, then the baselines, implementation details and hyperparameter tuning.

Evaluation Procedure As test set, we use the latest 10% of purchase events with associated past sessions. The remaining 90% is used for training. Since some users have had multiple purchase events, we remove purchase events from the training data, if their associated past sessions also appear in the test set.

The RS generates a prediction for what items the user will buy as a ranked list, i.e., items are ranked by their predicted probability: the closer to the top of the ranking, the higher the estimated probability of the item. There are two types of items: new insurance products and additional coverage. Since it is only possible for a user to buy an additional coverage if the user has the corresponding base insurance product, we use a post filter to set the probability to 0 if that is not the case, as per [2]. The resulting list of ranked items is evaluated with Hit Rate (HR), Mean Reciprocal Rank (MRR), Precision, Recall, and Mean Average Precision (MAP). We use a cutoff threshold of 3 because: (1) the total number of items is 16, therefore high cut-offs, e.g., ≥ 10, will increase recall and all measures will reach high values, which will not inform on the actual quality of the RSs; (2) on the user interface the user will be recommended up to 3 items. Additionally, we report HR and MRR scores for all cut-offs value from 1 to 5.

Experimental results are supported by statistical testing. For HR we use McNemar's test [37] and for all other measure we use one-way ANOVA [81], both with a confidence level of 0.05, and post hoc tests to control the family-wise error rate due to multiple comparisons.

[2] `https://github.com/simonebbruun/cross-sessions_RS`

Baselines We compare our models against the following state-of-the-art baselines:

- **Random** recommends random items to the user.
- **Popular** recommends the items with the largest number of purchases across users.
- **SVD** is a method that factorizes the user-item matrix by singular value decomposition [35]. The portfolio data forms the user-item matrix, where a user-item entry is 1 if the user has bought the item and 0 otherwise. Since in the insurance domain a user can buy the same item again (e.g., a second car insurance) and matrix factorization cannot be used for repeated recommendations, we add repeated items as new items (columns) to the matrix.
- **Demographic** is a classification model, as per [111, 112], that uses user demographics and their portfolios as input features. The portfolios are represented with a feature for each item counting how many of the items the user has already bought. Demographic features and portfolio features are concatenated. We use a feed forward neural network to make a fair comparison with the neural session-based approaches.
- **GRU4REC** is a neural session-based model [59, 60]. Its input is the last session of a user, consisting of the 3-tuple user actions described in Section 3.3.2. For every time step, the model outputs the likelihood for each action to be the one the user interacts with next. Recommendations are based on the output after the final time step.
- **GRU4REC Concat** is the same as GRU4REC, but all recent user sessions are concatenated into a single session.
- **SKNN_E** is a session-based nearest neighbour model as the one presented in [67] with the extension suggested in [83]. The nearest neighbours are determined based on the set of actions in all recent sessions of each user. A user's set of actions is a vector computed with a max pooling operation over all actions generated by the user (in recent sessions). We then adapt this baseline to our task, so the recommendations are based on the items purchased by the neighbours of the target user rather than the items interacted with in the ongoing session.
- **SKNN_EB** is the same as SKNN_E, but with a further extension suggested in [83]: scores of items previously interacted with are boosted with a factor. The factor is tuned as a hyperparameter.

Note that SVD and the demographic model make use of user portfolio, i.e., users past purchases, while this is not the case for all cross-sessions models, GRU4REC and SKNN. GRU4REC is included as it has shown best performance under identical conditions on various datasets among all the neural models compared in [97] and in [83].

We tried the sequential extension to SKNN, Vector Multiplication SKNN, which is presented in [96], but did not obtain better performance than the original one. Sequence and Time Aware Neighborhood [45] is not included as baseline since it was not possible to adapt it to the task under consideration (for the reasons discussed in Section 3.2.2).

Table 3.3: Hyperparameters (*autoencoder/RNN)

Model	Batch size	Units	Dropout
Demographic	32	32	0.3
GRU4REC	32	256	0.2
GRU4REC Concat	32	256	0.2
Cross-sessions Concat	128	64	0.3
Cross-sessions Encode	32	64	0.3
Cross-sessions Auto*	128/32	512/64	-/0.4

Implementation & Hyperparameters All implementation is in `Python 3.7.4` and `TensorFlow 2.6.0`[3]. We used `Adam` as the optimizer with TensorFlow's default settings for the learning rate, exponential decay rates and the epsilon parameter. Early stopping was used to choose the number of epochs based on the minimum loss on the validation dataset. We used two-layer networks[4] with dropout regularization on the first hidden layer.

We partitioned the training set in the same way as the whole dataset, so the validation set includes the latest 10% of purchases with associated sessions and the remaining is used for training. We tuned the hyperparameters of each neural model (batch size, number of units, and dropout rate) on the validation set using grid search. We test powers of 2 for the batch size and number of units ranging from 16 to 512. For the dropout rate, we test values in $[0.1, 0.5]$ with step size 0.1. The final hyperparameters used are reported in Tab. 3.3.

For GRU4REC and GRU4REC Concat we tried the 3 different loss functions: cross-entropy, BPR [116] and TOP1 [59]. Cross-entropy was finally chosen for both models, as it performed best on the validation set. For the non-neural models, the optimal number of latent factors for the SVD model was 1, the optimal number of neighbours for both SKNN_E and SKNN_EB was 30, and the optimal boost factor for SKNN_EB was 0.5. Neural models were trained on Nvidia GeForce MX250 equipped with 2GB of GPU memory. The maximum training time was 6 hours.

3.5.2 Experimental Results

Next we compare our cross-sessions against state-of-the-art baselines. Furthermore, we combine cross-sessions models with the demographic model. We further breakdown the analysis of our model to understand the impact of exploiting all past sessions and actions instead of only the most recent ones and their order. Lastly, we conduct an ablation study to show how different actions (sections, objects, and type) affect the performance of our models.

Performance Analysis Tab. 3.4 compares our cross-sessions models against the baselines. The simple Popular model is a quite strong baseline, unlike in domains like retail and video

[3]We used Tensorflow's implementation of padding and masking to deal with variable length input in the RNNs.

[4]In all models the second layer is a dense layer with ReLU activation function.

Table 3.4: Performance results. All results marked with * are significantly different from cross-sessions encode. The best score for each measure is in bold. Percentages in brackets denote the difference of our models from the strongest baseline (SKNN_EB).

Model	HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Random	0.3235*	0.1114*	0.2940*	0.1910*	0.1839*
Popular	0.6217*	0.2145*	0.5855*	0.4764*	0.4540*
SVD	0.6646*	0.2372*	0.6327*	0.4997*	0.4829*
Demographic	0.7392*	0.2649*	0.7095*	0.5620*	0.5446*
GRU4REC	0.6479*	0.2313*	0.6208*	0.5443*	0.5264*
GRU4REC Concat	0.6616*	0.2365*	0.6362*	0.5620*	0.5453*
SKNN_E	0.8106*	0.2914*	0.7848*	0.6740*	0.6567*
SKNN_EB	0.8132*	0.2922*	0.7872*	0.6785*	0.6610*
Cross-sessions Encode	**0.8380 (3.04%)**	**0.3030 (3.67%)**	**0.8145 (3.46%)**	**0.7093 (4.53%)**	**0.6923 (4.73%)**
Cross-sessions Concat	0.8265 (1.62%)	0.2984 (2.12%)	0.8019 (1.87%)	0.7051 (3.92%)	0.6876 (4.02%)
Cross-sessions Auto	0.8356 (2.74%)	0.3024 (3.48%)	0.8128 (3.24%)	0.7085 (4.41%)	0.692 (4.69%)

Table 3.5: The versions of our models enhanced with demographic data. The rest of notation is as in Tab. 3.4.

Model	HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode with Demographic	**0.8542* (5.03%)**	**0.3103* (6.17%)**	**0.8313* (5.6%)**	0.7268* (7.11%)	0.7099* (7.41%)
Cross-sessions Concat with Demographic	0.8497* (4.48%)	0.3087* (5.64%)	0.8269* (5.04%)	**0.7298* (7.55%)**	**0.7131* (7.88%)**
Cross-sessions Auto with Demographic	0.8460 (4.03%)	0.3072 (5.13%)	0.8228 (4.52%)	0.7223 (6.45%)	0.7050 (6.66%)

services [59, 44], because of the few different items and the role of the post filter to make sure not to recommend items that the user cannot buy.

As in prior work on insurance RSs [111], we also see a significant improvement in using a demographic RS compared to the traditional matrix factorisation method, SVD, due to the sparse feedback on items in the insurance domain (see Tab. 3.1) and users' demographic characteristics being good signals for learning insurance recommendations.

The session-based methods, SKNN_E, SKNN_EB and cross-sessions, significantly outperform the non-session-based methods, while this is not the case for GRU4REC and GRU4REC Concat. This shows that users' recent sessions are stronger signals for learning insurance recommendations than long-term preferences and demographic characteristics, but recommending the item that the user is most likely to interact with next on the website is not appropriate for insurance recommendations. All cross-sessions methods significantly outperform SKNN_E and SKNN_EB suggesting that an RNN is better at modeling relationships between the user actions that lead to the purchase of specific items.

The results suggest that encoding of sessions is better than the trivial concatenation of sessions indicating that dependencies across sessions are important. The results further suggest that the encoding of sessions with a max-pooling operation is better than the automatically learned encodings of sessions (using an autoencoder). This is most likely because the order of actions (which the autoencoder takes into account) adds more noise to the model than signal, or the autoencoder needs a larger amount of training data in order to effectively learn to encode

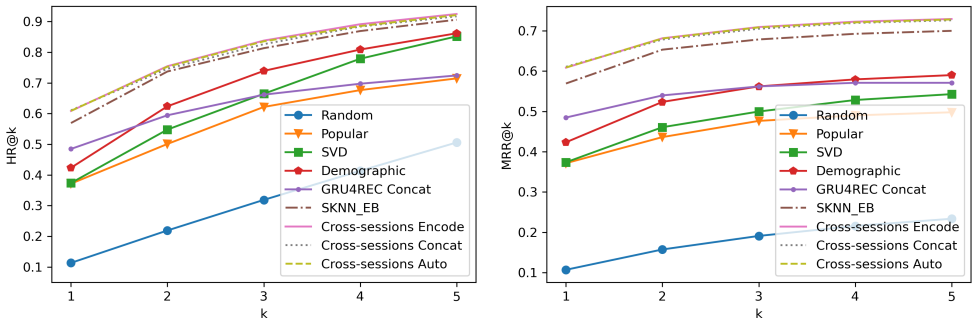


Figure 3.7: HR@k and MRR@k for varying choices of the cutoff threshold k.

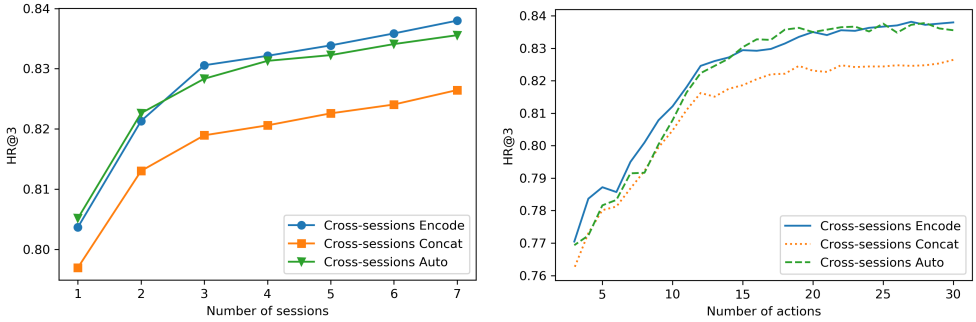


Figure 3.8: HR@3 for different number of recent sessions and actions.

sessions.

Fig. 3.7 shows HR and MRR at varying cutoffs k from 1 to 5. We have similar results for recall, precision, and MAP, which are omitted due to space limitations. The results are consistent over varying choices of the cutoff threshold, with the exception of GRU4REC Concat which is better than SVD and Demographic for smaller cutoff thresholds (1-2), but not for larger. Across all choices of the cutoff threshold there is a clear gap between cross-sessions models and the others. The general trend for both measures is that they tend to increase as the cut-off k increases. As expected, this happens because, by increasing the cutoff threshold, it is more likely to include the purchased items.

Finally, we combine our cross-sessions models with the demographics. The results are shown in Tab. 3.5. The hybrid approach between a cross-sessions and a demographic model yields better performance than the individual models for all models and evaluation measures. This indicates that the two types of information, sessions and demographic, capture different aspects of the problem. The best results are obtained with *Cross-sessions Encode* and *Cross-sessions Concat* and both models are statistically significantly different from the *Cross-sessions Encode* model without demographic data.

Table 3.6: Study of session order. Relative change in parentheses.

Model		HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode	original session order	0.8380	0.3030	0.8145	0.7093	0.6923
	shuffled session order	0.8345 (-0.41%)	0.3008 (-0.7%)	0.8096 (-0.60%)	0.7058 (-0.49%)	0.688 (-0.61%)
Cross-sessions Concat	original session order	0.8265	0.2984	0.8019	0.7051	0.6876
	shuffled session order	0.8209 (-0.20%)	0.2935 (-0.52%)	0.7925 (-0.37%)	0.6978 (-0.28%)	0.6759 (-0.49%)
Cross-sessions Auto	original session order	0.8356	0.3024	0.8128	0.7085	0.6920
	shuffled session order	0.8305 (-0.61%)	0.3003 (-0.70%)	0.8069 (-0.72%)	0.704 (-0.64%)	0.6867 (-0.76%)

Analysis of Number of Sessions and Actions Next, we analyse how the number of sessions affects our cross-sessions models. We break down the performance scores of our models based on the number of sessions, starting with only the most recent session, up to including all the available sessions (the maximum number of sessions per user is 7, see Sections 3.4.2 and 3.4.3). Fig. 3.8 (left) shows HR@3 computed for our cross-sessions models with respect to varying numbers of user sessions. In general, there is an increasing trend in performance with the number of recent sessions, emphasising the additional contribution brought by all recent sessions of each user rather than just the last one. We can observe that *Cross-sessions Encode* and *Cross-sessions Auto* consistently outperform *Cross-sessions Concat* for each number of sessions. We observe similar results for MRR, recall, precision, and MAP, which are not included here due to space limitations.

We do the same analysis with number of actions per session. Fig. 3.8 (right) shows that HR@3 generally increases with the number of actions. The growth is particularly steep up to about 10 actions per session after which it flattens out. We observe similar results for MRR, recall, precision, and MAP, which are not included here due to space limitations.

Analysis of Session Order We analyse the importance of session order by randomly shuffling the order of sessions and retraining the models. We shuffle the order in both training, validation and test data, and perform the experiment 5 times to account for randomness. The mean performance is presented in Tab. 3.6. Across all our models and evaluation measures, performance drops when shuffling the session order, but the decrease is limited to less than 1%. The results indicate that the superiority of the cross-sessions models is not due to sequential dependencies, rather they are simply better at capturing the relationships between user actions and the purchase of specific items.

Analysis of Actions We use ablation to analyse the influence of different actions, i.e., sections, objects, and types. Each time we remove all actions of a given type and evaluate our cross-sessions models after re-training without the action type under analysis. The results are presented in Tab. 3.7 for all our models and evaluation measures. We did not consider the action type "click" in the ablation study because removing clicks results in removing most of the actions (65%), but also most of the objects and sections, since users interact with objects and sections mainly through clicks.

Table 3.7: Ablation study of actions. Relative change in parentheses.

Model		HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode	all actions	0.8380	0.3030	0.8145	0.7093	0.6923
	without E-commerce	0.7526 (-10.19%)	0.2698 (-10.95%)	0.7249 (-11.00%)	0.5951 (-16.09%)	0.5764 (-16.74%)
	without Claims reporting	0.8250 (-1.55%)	0.2979 (-1.68%)	0.8012 (-1.64%)	0.7006 (-1.22%)	0.6829 (-1.35%)
	without Information	0.8317 (-0.75%)	0.3000 (-0.98%)	0.8072 (-0.89%)	0.7045 (-0.68%)	0.6863 (-0.87%)
	without Personal account	0.8067 (-3.73%)	0.2894 (-4.48%)	0.7803 (-4.19%)	0.6604 (-6.89%)	0.6438 (-7.00%)
	without Items	0.7379 (-11.94%)	0.2652 (-12.46%)	0.7116 (-12.63%)	0.5720 (-19.36%)	0.5548 (-19.86%)
	without Services	0.8032 (-4.15%)	0.2880 (-4.93%)	0.7765 (-4.66%)	0.6639 (-6.40%)	0.6465 (-6.61%)
	without Start	0.8162 (-2.60%)	0.2935 (-3.11%)	0.7906 (-2.94%)	0.6771 (-4.55%)	0.6592 (-4.77%)
	without Act	0.8318 (-0.73%)	0.3005 (-0.80%)	0.8082 (-0.77%)	0.7035 (-0.82%)	0.6864 (-0.85%)
	without Complete	0.8317 (-0.75%)	0.3005 (-0.82%)	0.8078 (-0.82%)	0.7036 (-0.80%)	0.6861 (-0.89%)
Cross-sessions Concat	all actions	0.8265	0.2984	0.8019	0.7051	0.6876
	without E-commerce	0.7456 (-9.79%)	0.2677 (-10.30%)	0.7188 (-10.37%)	0.5918 (-16.08%)	0.5723 (-16.77%)
	without Claims reporting	0.8287 (+0.27%)	0.2987 (+0.08%)	0.8037 (+0.22%)	0.7067 (+0.22%)	0.6879 (0.04%)
	without Information	0.8308 (+0.53%)	0.3009 (+0.81%)	0.8076 (+0.71%)	0.7133 (+1.15%)	0.6958 (1.19%)
	without Personal account	0.8130 (-1.63%)	0.2914 (-2.35%)	0.7872 (-1.84%)	0.6795 (-3.64%)	0.6633 (-3.52%)
	without Items	0.7348 (-11.10%)	0.2630 (-11.87%)	0.7077 (-11.76%)	0.5783 (-17.99%)	0.5555 (-19.21%)
	without Services	0.8119 (-1.76%)	0.2914 (-2.37%)	0.7856 (-2.04%)	0.6798 (-3.59%)	0.6635 (-3.50%)
	without Start	0.8162 (-1.24%)	0.2943 (-1.39%)	0.7913 (-1.32%)	0.6851 (-2.84%)	0.6676 (-2.90%)
	without Act	0.8304 (+0.47%)	0.3001 (+0.56%)	0.8063 (+0.55%)	0.7083 (+0.44%)	0.6908 (0.47%)
	without Complete	0.8276 (+0.14%)	0.2995 (+0.35%)	0.8038 (+0.23%)	0.7065 (+0.19%)	0.6912 (0.91%)
Cross-sessions Auto	all actions	0.8356	0.3024	0.8128	0.7085	0.6920
	without E-commerce	0.7473 (-10.57%)	0.2691 (-11.02%)	0.7206 (-11.34%)	0.5953 (-15.97%)	0.5776 (-16.53%)
	without Claims reporting	0.8227 (-1.54%)	0.2966 (-1.93%)	0.7979 (-1.83%)	0.7001 (-1.19%)	0.6819 (-1.45%)
	without Information	0.8287 (-0.82%)	0.2999 (-0.82%)	0.8048 (-0.99%)	0.7055 (-0.42%)	0.6881 (-0.56%)
	without Personal account	0.8175 (-2.17%)	0.2925 (-3.27%)	0.7916 (-2.60%)	0.6809 (-3.89%)	0.6639 (-4.07%)
	without Items	0.7398 (-11.46%)	0.2649 (-12.41%)	0.7110 (-12.52%)	0.5771 (-18.55%)	0.5585 (-19.29%)
	without Services	0.8051 (-3.64%)	0.2892 (-4.37%)	0.7792 (-4.13%)	0.6741 (-4.86%)	0.6574 (-5.00%)
	without Start	0.8183 (-2.07%)	0.2942 (-2.71%)	0.7924 (-2.50%)	0.6842 (-3.43%)	0.6661 (-3.74%)
	without Act	0.8287 (-0.82%)	0.2993 (-1.03%)	0.8047 (-1.00%)	0.6994 (-1.28%)	0.6817 (-1.49%)
	without Complete	0.8257 (-1.18%)	0.2987 (-1.23%)	0.8025 (-1.27%)	0.7024 (-0.85%)	0.6852 (-0.98%)

For the *Cross-sessions Encode* and *Cross-sessions Auto* models, all actions contribute positively to the model performance, i.e., their removal causes a drop in the measure score. We can have a similar conclusion for the *Cross-sessions Concat* model with the exception that the removal of actions in the section "information" and "claims reporting", or of type "act" and "complete" increases performance instead of decreasing it, but the actual difference is negligible (less than 1.5%). The *Cross-sessions Concat* model is more prone to overfitting when including these actions with weaker preference signals because it estimates weights for every action and takes the order of actions into account while e.g., the *Cross-sessions Encode* model only accounts for the order of sessions.

Not surprisingly, the most negative impact (up to -19.86% in MAP) is obtained when actions with the object type "item" are removed. Even if these are not the most frequent objects (see Tab. 3.2), they are highly informative as they provide information on the user interests. The most frequent objects are "services", twice more frequent than items, and their removal affects negatively performance even if with a less severe impact (up to -6.61% in MAP).

The second greatest negative impact is obtained when the actions from the section "e-commerce" are removed (up to -16.77% in MAP). Again, this is not the most frequent section, but it is highly informative since the user needs to access the e-commerce section to inspect and compare different insurance products. The most frequent section is the "personal account", 3

times more frequent than the e-commerce section. Its removal has a negative impact, but not as severe as for e-commerce (up to -7% in MAP).

In terms of action type, the removal of the type "start" has the greatest negative effect, even if limited with respect to the other two categories (up to -4.77% in MAP). The types "act" and "complete" have a negligible impact (less than 1.5%), but they also represent a very sparse signal (less than 5% of all types together).

3.6 Conclusions and Future Work

We have tackled the problem of recommendation of insurance products. This is a highly relevant industrial problem, to which no satisfying solution currently exists, because of the low number of items and sparsity of user interactions. We propose three different cross-sessions recommendation models, which exploit both the current and past user sessions to predict items that the user will buy. Our models take as input an ordered sequence of sessions each being a list of actions and model the dependencies across sessions using RNNs with GRU units. Experimental results on a real world dataset show that our models outperform state-of-the-art baselines. Further analysis confirms the positive effect of considering multiple past sessions and an ablation study shows that all considered action types are beneficial for the models. Demographic features boost performance of the cross-sessions model, giving rise for future work on potential biases of demographic insurance recommendations.

As future work we further plan to run an A/B testing experiment to evaluate our cross-sessions models with online users. Furthermore, we will investigate the explainability of our models and we will generate user readable explanations to be shown when an item is recommended.

Acknowledgements

This paper is partially supported by the EU Horizon 2020 MSCA grant No. 893667.

Chapter 4

Recommending Target Actions Outside Sessions in the Data-poor Insurance Domain

Abstract

Providing personalized recommendations for insurance products is particularly challenging due to the intrinsic and distinctive features of the insurance domain. First, unlike more traditional domains like retail, movie etc., a large amount of user feedback is not available and the item catalog is smaller. Second, due to the higher complexity of products, the majority of users still prefer to complete their purchases over the phone instead of online. We present different recommender models to address such data scarcity in the insurance domain. We use recurrent neural networks with 3 different types of loss functions and architectures (cross-entropy, censored Weibull, attention). Our models cope with data scarcity by learning from multiple sessions and different types of user actions. Moreover, differently from previous session-based models, our models learn to predict a target action that does not happen within the session. Our models outperform state-of-the-art baselines on a real-world insurance dataset, with ca. 44K users, 16 items, 54K purchases and 117K sessions. Moreover, combining our models with demographic data boosts the performance. Analysis shows that considering multiple sessions and several types of actions are both beneficial for the models, and that our models are not unfair with respect to age, gender and income.

4.1 Introduction

We present the problem of providing automatic personalised recommendations in the insurance purchasing domain. That is to recommend insurances for individuals such as home insurance, car insurance and accident insurance that can help customers of an insurance company continuously

adjust their insurances to suit their needs. The following reasons make this problem particularly challenging: (i) the item catalog is small (only a few types of insurance products are available for purchase), unlike the common scenarios of e-commerce or movie recommendation, where the item catalog is very large; (ii) insurance products are purchased less often and tend to last longer, than items commonly bought in common recommendation scenarios; (iii) most users prefer to complete their purchases of insurance products in a telephone conversation with a human insurance agent who can interactively address their concerns, instead of fully and exclusively online. Collectively, the above reasons result in a limited amount of user interactions and feedback on the items available for recommendation, making the problem of automatically learning recommendations non-trivial.

To address this problem, we present three different architectures of recurrent neural network recommender models, which deal with the data scarcity problem outlined above in two ways. Firstly, they learn user preferences not only from single sessions and from user actions that are associated with an item, but from multiple past user sessions and several different types of user actions that are not necessarily associated with an item. Secondly, unlike existing session-based and session-aware recommender models, our models learn to predict a target action (purchase of an insurance item) that does not happen within the input session (because the purchase happens on the telephone conversation with a human insurance agent, for instance).

We show experimentally that these two features of our models make them outperform state-of-the-art recommender baselines on a real-world insurance dataset of ca. 44K users, 16 items, 54K purchases and 117K sessions (which is freely available to the research community).

This paper is an extended version of the conference full paper [20] accepted at RecSys 2022. We **contribute** by extending the previous work with (1) two new approaches to design the loss function and architecture of the proposed cross-sessions recommender system; (2) an evaluation of the novel approaches against the state-of-the-art and the original cross-session recommender system; (3) additional analysis (e.g., fairness analysis, analysis of the dependency on the temporal threshold); and (4) more details on the exploited dataset.

4.2 Related Work

We present prior work on insurance recommendation (Section 4.2.1) and session-based recommendation (Section 4.2.2).

4.2.1 Insurance Recommendation

There is not much prior work on insurance recommendations. In principle, knowledge-based recommendations should work for this task by mining highly personalised user information from user interactions [65], but to our knowledge, no prior work reports this. Instead, most prior work on insurance recommendations supplements the small volume of user feedback with user demographics. We overview these next.

Xu et al. [161] cluster users based on their demographics and make association rule analysis within each cluster on the users' set of purchased items. They extract recommendations directly from the association rules. Sanghamitra Mitra [122] estimate user similarity based on demographic attributes using a similarity measure (e.g., cosine similarity). Then they make recommendations to a user based on the feedback on items by the top-N similar users. Qazi et al. [111, 112] train a Bayesian network with user demographics and previously purchased items as input features, aiming to predict the last purchased item of a user. They also train a feed-forward neural network to give recommendations to potential users when only external marketing data is available. All the above methods outperform standard RS approaches, such as matrix factorization and association rule mining solely applied to the feedback data. In addition, these methods are not susceptible to cold start issues when recommending items to users with no previous feedback on items [50]. However, the above methods assume that the preference dynamics are homogeneous within demographic segments, which is not necessarily true. Our model addresses this by accommodating individual changes in user preferences through the use of sessions generated by the individual user.

Bi et al. [14] present a cross-domain approach for insurance recommendation. They use knowledge from an e-commerce domain (clothes, skincare products, fruits, electronics products, etc.) to learn better recommendations in the insurance domain when data is sparse. They employ a Gated Recurrent Unit (GRU) [34] to model sequential dependencies in the e-commerce domain. Our model differs from this approach: we use user sessions directly from the insurance target domain instead of another source domain like an e-commerce website with clothes etc., thereby not having the need for overlapping users between the target domain and a source domain. Moreover, we cannot use the approach of [14] in our work because it is not session-based; it is based on users' long-term preferences in both the e-commerce and the insurance domain.

4.2.2 Session-based Recommender Systems

Session-based RSs capture the user's short-term preferences in a session [43], often using item-to-item recommendations [36, 90]: similarities between items are computed based on the session data such that items that often co-occur and/or co-interact in a session fetch a high similarity. Such item similarities are then used during a session to recommend the most similar items to the item that the user currently interacts with. This approach only considers the last user interaction; it ignores information on past interactions even in the same session. Moreover, this approach requires the target action to happen within sessions, thus this approach cannot be applied to our task.

An extension of the item-to-item approach is session-based clustering, which considers all user interactions in the session. Sessions are then clustered in various ways, for instance as Markov chains [51], or using K-Nearest Neighbors (SKNN) [67, 62], which computes similarities between entire sessions using a similarity measure (e.g., cosine similarity). The recommendations are then based on selecting items that appeared in the most similar past session. This approach does not take into account the order of the input sequence.

A sequential extension to the SKNN method is Vector Multiplication SKNN [96], which

rewards the most recent user interactions of a session when computing the similarities. Another extension is Sequence and Time Aware Neighborhood [45], which considers the position of an item in the current session, the recency of a past session with respect to the current session, and the position of a recommendable item in a neighbouring session. The latter model depends on the target actions occurring within sessions, and so does not fit our task where the target actions occur outside the session.

Neural session-based methods have also been used. A popular one is GRU4REC [59, 60], which models user sessions with a GRU in order to predict the probability of the subsequent interaction given the sequence of previous interactions. Neural Attentive Session-based Recommendation [88] extends GRU4REC with an attention mechanism that captures the user's main purpose in the current session. Graph Neural Networks have also been used [156] to model session sequences as graph structures, thereby capturing transitions of items and generating item embedding vectors correspondingly.

The above methods use only the single ongoing session of a user, whereas we use multiple past sessions of a user. Methods that use past user sessions when predicting the next interaction for the current session have been proposed [114, 120, 164, 63, 110]. However, a comprehensive empirical study of these methods [83] shows that they did not improve over heuristic extensions of existing session-based algorithms that for instance extend the current session with previous sessions or boost the scores of items previously interacted with. Unlike all the above methods, we use different types of actions, not only with items (see Section 4.4.1).

Although session-based RSs provide temporal context for the recommendations in terms of the sequential order of interactions, they do not account for the actual time of interactions. Other types of RSs exploit time as a contextual variable in the learning of recommendations [130, 18]. However, these methods use the specific property of time at the level of specificity of the hour, day, week, month or season to learn the recommendations. Other RSs integrate methods from survival analysis to predict users' return time to a service [42, 73] that for instance can help in planning advertising inventories. Our approach to account for time in the RS differs from these, since we do not focus on a single return time to estimate when a user comes back, rather we model times for each item to be used for the ranking task.

4.3 Approach

In this section, we present the problem formulation (Section 4.3.1) and how we address it with our approach (Section 4.3.2).

4.3.1 Problem Formalization

The task of our cross-sessions RS is to predict what items a user will buy based on the user's past sessions. Compared to the traditional task of a session-based RS, which is to predict the next action in the session based on the actions so far, our task differs because: (1) the target action (i.e., purchase) occurs outside the session; (2) the user may have multiple sessions leading

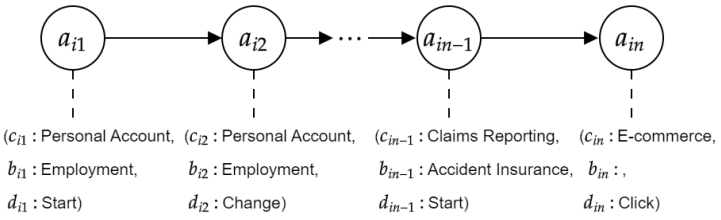


Figure 4.1: Example of a session on the insurance website. A session is a list of 3-tuple actions ordered by time.

to the purchase; (3) the sessions consist of many different actions, not only actions with items. We extend the notation in Fang et al. [43] to accommodate these differences when formulating the problem.

A session, s_i, is a sequence of user actions, $\{a_{i1}, a_{i2}, a_{i3}, ..., a_{in}\}$, on the website. An action, a_{ij}, is represented by the 3-tuple $a_{ij} = (c_{ij}, b_{ij}, d_{ij})$, where:

- c_{ij}: action section, refers to the section of the website in which the user interacts;
- b_{ij}: action object, refers to the object on the website that the user interacts with; and
- d_{ij}: action type, refers to the way that the user interacts.

Figure 4.1 illustrates a session where a user interacts with employment (object b_{i1}) at the personal account section of the website (section c_{i1}) by starting it (action type d_{i1}). Then, the user changes (d_{i2}) the employment (b_{i2}) still at the personal account section (c_{i1}). Section 4.4.1 and Table 4.2 present the different sections, objects and action types.

We represent users' past sessions in lists ordered by time. We do not include all historical sessions of a user as we assume that only recent sessions are relevant to the current task. We use an inactivity threshold, t, to define recent sessions, and define two sessions to belong to the same task if there is no longer than the threshold t between them. We describe how we estimate t from observed data in Section 4.4.1. The problem is to learn a function, f, of a user's session sequence that estimates the probability of the user to buy each item k after the last session s_m:

$$f(s_1, s_2, s_3, ..., s_m) = (\hat{p}_1, \hat{p}_2, \hat{p}_3, ..., \hat{p}_K), \tag{4.1}$$

where each element in $\{s_1, s_2, s_3, ..., s_m\}$ is a session (defined as a sequence of actions as explained above), $\hat{p}_k$ is the predicted probability that item k will be bought by the user, and K is the total number of items in the whole dataset.

4.3.2 Proposed Approach

Our model for learning the above function is based on GRU4REC [59]: an RNN with a single GRU [34] layer that models user interactions with items in single user sessions. The RNN takes

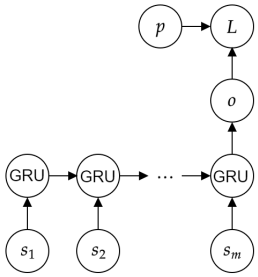


Figure 4.2: Architecture of cross-sessions encode.

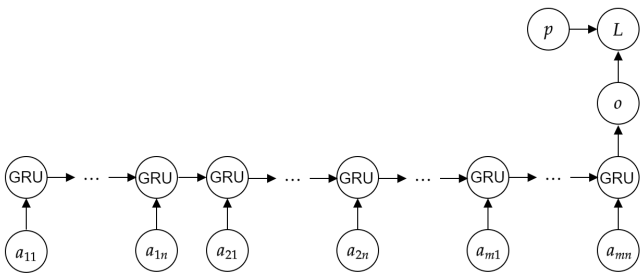


Figure 4.3: Architecture of cross-sessions concat.

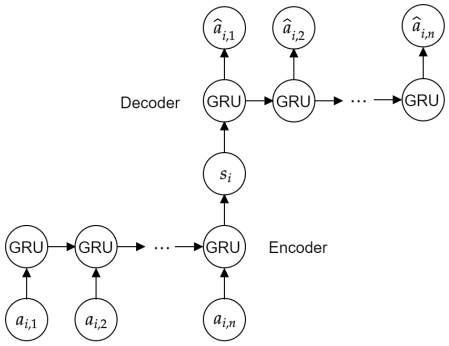


Figure 4.4: Architecture of cross-sessions auto.

as input the ordered sequence of items interacted with in the session, and outputs for every time step the likelihood of each item to be the item that the user interacts with next. Our cross-sessions RS extends GRU4REC by: (1) taking multiple sessions of each user as input, as in Eq. (4.1); (2) using various types of input actions that are not always associated with items; (3) predicting what items the user will buy after the last time step as opposed to predicting the next interaction for every time step in the sequence.

Model input Next, we propose three different ways of passing the input sessions through the RNN.

In the first way, which we call *Cross-sessions Encode* (see Figure 4.2), we encode a session by aggregating the actions in the session with a maximum pooling operation:

$$s_i = \max_{element}(a_{i1}, a_{i2}, a_{i3}, ..., a_{in}), \tag{4.2}$$

where a_{ij} is the binarized vector indicating the presence of an action section, action object and

action type performed by a user at time step j in session i, and $\max_{element}(\cdot)$ is a function that takes the element-wise maximum of vectors. Then, for every time step i in the sequence of a user's sessions, an RNN with a single GRU layer computes the hidden state as follows:

$$\begin{aligned} h_i &= (1 - z_i) \cdot h_{i-1} + z_i \cdot \hat{h}_i && \text{for } \; i = 1, .., m, \\ z_i &= \sigma(W_z s_i + U_z h_{i-1}), && (update \;\; gate) \\ \hat{h}_i &= \tanh(W s_i + U(r_i \cdot h_{i-1})), && (candidate \;\; gate) \\ r_i &= \sigma(W_r s_i + U_r h_{i-1}), && (reset \;\; gate) \end{aligned} \tag{4.3}$$

where W_z, U_z, W, U, W_r and U_r are weight matrices and $\sigma(\cdot)$ is the sigmoid function. The reset gate makes sure to forget information about the past that is not important given the current session. The update gate decides whether the current session contains relevant information that should be stored. The hidden state, h_i, is a linear interpolation between the previous hidden state and the candidate gate.

Our second way of passing input sessions through the RNN is called *Cross-sessions Concat* (see Figure 4.3). Here, we concatenate all sessions of a user into a single sequence $s = \{a_{11}, .., a_{1n}, a_{21}, .., a_{2n}, .., a_{m1}, .., a_{mn}\}$. Now the hidden state in Equation (4.3) is computed for every time step (ij) in s. Whereas the cross-sessions encode only accounts for the order of sessions, the cross-sessions concat further takes into account the order of actions.

Finally, we propose another variation of session encoding, *Cross-sessions Auto* (see Figure 4.4), which automatically learns encodings of sessions with an autoencoder, instead of Equation (4.2). We train an RNN-based autoencoder with a single GRU layer that takes as input the ordered sequence of actions in a session and is evaluated on the task of recreating the input using categorical cross-entropy loss on each of the 3 features: action section, action object and action type. Once trained, the encoder is used to encode a session into a single vector, s_i, that can be used as input for Equation (4.3). Hence, the architecture of the autoencoder in Figure 4.4 is combined with the architecture in Figure 4.2.

Cross-Entropy Loss In all three cases, the RNN returns an output vector, o, of length K after the last time step. Because a user can buy multiple items at the same time, we consider the learning task as multi-label classification and use the sigmoid function, $\sigma(\cdot)$, on each element of o as output activation function to compute the likelihood of purchase:

$$\hat{p}_k = \sigma(o_k), \quad \text{for } \; k = 1, \ldots, K. \tag{4.4}$$

During training, the loss function is computed by comparing $\hat{p}$ with the binarized vector of the items purchased, p. Due to the learning task being multi-label classification, we define the loss function as the sum of the binary cross-entropy loss over all items. The loss function is thereby different from the ranking loss used in GRU4REC and is given by:

$$L = -\sum_{k=1}^{K} p_k \cdot \log(\hat{p}_k) + (1 - p_k) \cdot \log(1 - \hat{p}_k). \tag{4.5}$$

Censored Weibull Loss Because users often have multiple sessions before they purchase, and at the time of prediction we do not know when they will purchase, we furthermore propose another loss function that takes the time of purchase into account. Instead of predicting the probability of each item being purchased after the last time step, we now predict the time to the next purchase (time-to-purchase) of each item after every time step in the input sequence of the user's sessions, where time is measured in days. The loss function should then compare the predicted time-to-purchase with the true time-to-purchase. However, only some of the true times are observed, since a user typically has not purchased all of the items by the end of the training period. This is what is called censored data since it is only partially observed. For that reason, we use the loss function presented in Martinsson [99] that takes into account censored time data.

Let $y_{i,k}$ be the observed time-to-purchase of item k at time step i. Moreover, let $u_{i,k}$ be the censoring variable that indicates whether the purchase of item k has occurred after time step i and before the end of the training period. We assume that $y_{i,k}$ is a realisation of the random variable $Y_{i,k}$ that follows a discrete Weibull distribution with positive parameters $\alpha_{i,k}$ and $\beta_{i,k}$ and probability mass function given by:

$$P(Y_{i,k} = y_{i,k}) = \exp\Big(-\Big(\frac{y_{i,k}}{\alpha_{i,k}}\Big)^{\beta_{i,k}}\Big) - \exp\Big(-\Big(\frac{y_{i,k}+1}{\alpha_{i,k}}\Big)^{\beta_{i,k}}\Big), \text{ for } y_{i,k} = 0, 1, 2, ... \tag{4.6}$$

We use the Weibull distribution as it is commonly used to model time-to-event data because it is positive, has infinite support, and contrary to other time-to-event distributions, like the exponential distribution and log-logistic distribution, the Weibull distribution has a discrete version, that can be used when time is measured in, for example, days. The RNN returns two output vectors o_i^1 and o_i^2, both of length K for every time step $i = 1, ..., m$, which are then transformed with an output activation function into valid parameters of the Weibull distribution (i.e., positive values). We use an exponential activation function to compute $\alpha_{i,k}$ as it has shown to give fast training time in Chen et al. [32] due to logarithmic effect of change in $\alpha_{i,k}$, and we use a sigmoid activation function to compute $\beta_{i,k}$ as we want slow changes when $\beta_{i,k}$ gets close to 0:

$$\begin{pmatrix}\alpha_{i,k}\\ \beta_{i,k}\end{pmatrix} = \begin{pmatrix}\exp(o_{i,k}^1)\\ \sigma(o_{i,k}^2)\end{pmatrix}, \text{ for } k = 1, ..., K \text{ and } i = 1, ..., m. \tag{4.7}$$

The loss function is given by:

$$L_i = -\sum_{k=1}^{K} u_{i,k} \log\Big(P(Y_{i,k} = y_{i,k})\Big) + (1 - u_{i,k}) \log\Big(P(Y_{i,k} > y_{i,k})\Big), \text{ for } i = 1, ..., m, \tag{4.8}$$

where $P(Y_{i,k} > y_{i,k})$ is the right tail probability given by:

$$P(Y_{i,k} > y_{i,k}) = \exp\Big(-\Big(\frac{y_{i,k}+1}{\alpha_{i,k}}\Big)^{\beta_{i,k}}\Big). \tag{4.9}$$

Thus the loss function is given by the negative log-likelihood, where the likelihood is defined as the probability mass under the estimated parameters $\alpha_{i,k}$ and $\beta_{i,k}$ when the true time-to-purchase of item k is uncensored ($u_{i,k} = 1$), and the likelihood is given by the probability of

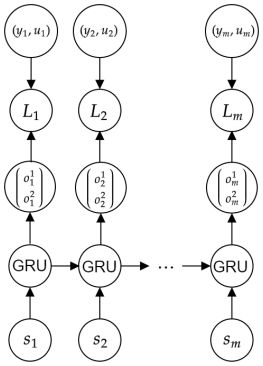


Figure 4.5: Architecture of Cross-sessions Encode with censored Weibull loss.

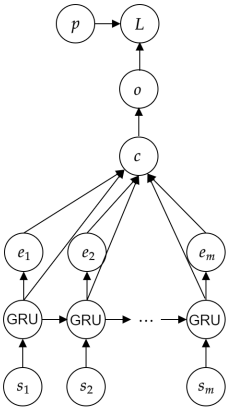


Figure 4.6: Architecture of Cross-sessions Encode with attention mechanism.

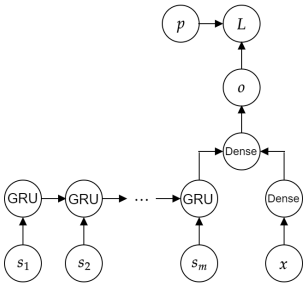


Figure 4.7: Architecture of a hybrid model between a cross-sessions and a demographic model. x denotes an input vector representing demographic features of the user.

the purchase to occur at some point after the end of training period when the true time-to-purchase is censored ($u_{i,k} = 0$). Finally, the loss is summed up over all items. The architecture of Cross-sessions Encode with censored Weibull loss is illustrated in Figure 4.5. The architecture of Cross-sessions Concat is similar, but the input is as in Figure 4.3 and the censored Weibull loss is computed for every action instead of every session. The architecture for Cross-sessions Auto with censored Weibull loss is the same as the one for Cross-sessions Encode, but combined with the architecture in Figure 4.4 to encode the input sessions. In the recommendation phase, the time-to-purchase for each item is computed as the median of the Weibull distribution under the predicted $\alpha_{i,k}$ and $\beta_{i,k}$ parameters. The score for each item is then given by the negative time-to-purchase, such that the shorter time-to-purchase of an item the higher the score.

Attention Model We propose to extend our cross-sessions models with an attention mechanism to account for the importance of different time steps in the input sequence. For instance, more weights could be given to more recent input passed to the RNN. We do this by adding an attention mechanism on the top of the GRU layer in our models that automatically learns attention weights. We use the Bahdanau attention introduced in Bahdanau et al. [9] to permit the decoder of the network to use the most relevant parts of the input sequence, by a weighted combination of all the hidden states, with the most relevant states being given the highest weights. Formally, instead of returning an output vector o after the last time step in the recurrent layer, the RNN returns attention scores e_i for each time step i. The attention scores are

then normalised into attention weights, λ_i, using a softmax function:

$$\lambda_i = \frac{\exp(e_i)}{\sum_i \exp(e_i)}. \tag{4.10}$$

Subsequently, a context vector c is computed as the sum of all the hidden states weighted by the attention weights:

$$c = \sum_i \lambda_i h_i. \tag{4.11}$$

The output vector o of length K is now returned from this attention layer and activated with the sigmoid function as in Equation 4.4. The architecture of Cross-sessions Encode with attention mechanism is illutrated in Figure 4.6, where the loss function is cross-entropy as in Equation (4.5).

Hybrid Model Finally, we propose a hybrid of a cross-sessions and a demographic model where the hidden state from the cross-sessions' RNN is merged with the hidden state from a feed-forward neural network with demographic input features of the user. This concatenation is then passed through a dense layer. The architecture of Cross-sessions Encode combined with demographics is illustrated in Figure 4.7, where the loss function is cross-entropy as in Equation (4.5). The combination with demographics is similar in the other cases.

4.4 Dataset

To the best of our knowledge, there is only one publicly available dataset that satisfies the criteria of our set-up [20], specifically: (1) item scarcity; (2) target action happening outside the session; and (3) different types of actions which might not be directly associated to an item or a purchase event. In the following, we describe such dataset (Section 4.4.1), we present how to estimate the temporal threshold to deem two sessions as belonging to the same task (Section 4.4.2), and we describe data pre-processing (Section 4.4.3).

4.4.1 Dataset Description

We use a publicly available dataset for the insurance domain[1]. Table 4.1 reports overall statistics about the dataset. The dataset consists of user logs on an insurance vendor website collected between October 1, 2018 and September 30, 2020. During this period no recommender system was implemented on the insurance website that could have affected the user behavior (e.g., exposure bias or position bias [29]). The dataset includes interaction logs of 44K users, who were uniquely identified either by log-in or cookies. There is a total of 16 items, including insurance products or additional insurance coverages. An additional coverage can be bought only if the customer already has the corresponding base product. There are around 53K purchase events

[1] `https://github.com/simonebbruun/cross-sessions_RS/tree/main/extended`

Table 4.1: Main properties of the dataset (*mean/std).

Users	44,434
Items	16
Purchase events	53,757
Sessions	117,163
Actions	1,256,156
Purchase events per user*	1.21/0.51
Sessions before purchase event*	2.18/1.68
Actions per session*	10.72/7.85

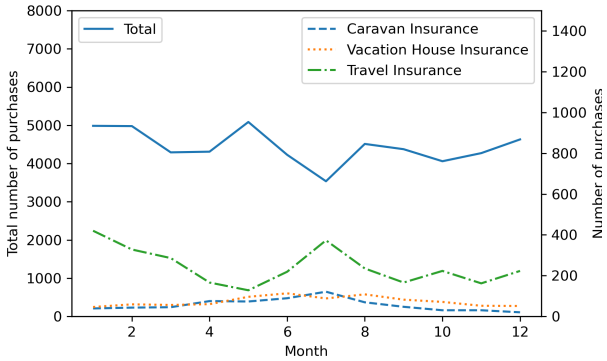


Figure 4.8: Number of purchases per month for all items (total) and for three selected items (caravan, vacation house and travel insurance).

that happened online or over the phone. Around 75% of the users prefer to complete their purchases over the phone instead of online. Figure 4.8 shows that there is no strong seasonal effect over the purchase frequency that the model should account for. Even caravan insurance, vacation home insurance and travel insurance have only minor seasonal trends.

Note that this dataset differs from other publicly available datasets typically used to evaluate session-based RSs, Last.fm, Recsys Challenge (RSC) datasets, etc. First, the total number of items is 16, much lower than the 91K items in Last.fm or the 29K items in RSC15. Second, insurance customers interact less frequently with the insurance website, indeed there are on average 1.2 purchases per user and 2.2 sessions before each purchase over a period of 2 years. Third, due to this scarcity of user interactions, the dataset is a collection of not only clicks and purchases in connection with items, but all types of actions, even if they are not directly associated with items. Therefore, this is one of the few publicly available datasets which contains several types of actions.

For each user-purchase pair, all sessions occurring before the purchase event are collected. This corresponds to around 100K sessions. A session (see Section 4.3.1) is represented as a

Table 4.2: Actions included in the dataset with their description and frequency.

Action section	E-commerce	Includes all products and their information, description, price, etc. Users can purchase products in this section.	256,319 (20.41%)
	Claims reporting	Contains a form where users can report claims.	11,188 (0.89%)
	Information	Information about e.g., payment methods and contact details.	198,147 (15.77%)
	Personal account	Users can log in and see their current insurances and claims in progress, change their personal information and adjust e.g., deductibles of current insurances.	790,502 (62.93%)
Action object	Items	Insurance product or additional coverage.	249,378 (19.85%)
	Services	All objects that are not items, e.g., payment methods and personal information.	655,574 (52.19%)
	No object	Click, without any object, e.g., a tab, drop down menu or front pages.	351,204 (27.96%)
Action type	Click	Click on the webpage.	811,747 (64.62%)
	Start	Start of e.g., purchase or claims report.	388,215 (30.90%)
	Act	Add product to cart, fill out claims report, etc.	47,713 (3.80%)
	Complete	Complete e.g., purchase or claims report.	8,481 (0.68%)

sequence of actions sorted by their timestamp, where each action has 3 different components:

- Action section: e-commerce, claims reporting, information and personal account;
- Action object: item and service:
- Action type: click, start, act and complete.

Table 4.2 provides an overview of the actions and their frequency in the dataset. Most of the actions (63%) occur in the personal account section because the users are existing customers, and most of them log in to their personal account. The second most frequent section is the e-commerce section (20%) because items are displayed in this section and sessions are collected before purchase events. Among the objects, the most frequent are services (52%). The insurance website allows the user to access a number of administrative services, for example, specifying or updating the employment type, this is needed for accident insurances, or specification of annual mileage, required by car insurances, or information about the insurance coverage when moving to a new house. Actions can happen without a specified object, in this case, they are denoted as "no object" (30%). This happens when a user interacts with a section, but without an object in the section, for example when a user enters the front page of a section. Clicks are the most frequent action type (64%). This does not surprise as clicks are the primary mean of interaction with a website. The second most frequent action is "start" (31%), which occurs when a user starts for example a purchase or a claims report. The action types "act" and "complete" are rather infrequent (less than 5% together). Examples of actions "act" are "change", when a user changes the employment type on the personal account page, or "fill out" when a user fills a form to report a claim. The action "complete" occurs when a user completes a change, for example, when a user completes the change of employment or the report of a claim.

Besides user actions and purchases, the dataset includes additional features, namely demographic attributes and portfolios of users. Demographic attributes are aggregated at a geographic area level, meaning they represent the average attribute of people living in the same area and not the exact value for every single user. The dataset includes the following demographic attributes: age, employment, income, residence, marital status, children and education. User portfolios include the purchase history of users within the insurance company, that is all items bought by each user. Our cross-sessions models exploit only user interactions (sessions and purchases) and do not use the demographic attributes or the user portfolios. However, these are needed by some state-of-the-art RS models, as SVD or the demographic model (see Section 4.5.1).

4.4.2 Estimation of Session Threshold

We consider a user task as a sequence of sessions that belong to the same task, that is the same user need, which eventually concludes in a purchase. Next, we explain how to deem two sessions as belonging to the same task. We need to estimate a temporal threshold t, such that given two subsequent sessions s_1 and s_2, if:

$$\texttt{start\_time}(s_2) - \texttt{start\_time}(s_1) \leq t \implies s_1 \text{ and } s_2 \text{ belong to the same task.} \quad (4.12)$$

That is, if the elapsed time between s_1 and s_2 is lower or equal to t, we will consider the two sessions as belonging to the same task.

This problem is similar to the one of estimating session boundaries for web users. It is straightforward to get the starting time of a user session, but it is not always possible to get the end time, for example, whenever a user leaves a browser window open and goes ahead with other non-related tasks, for example checking emails [71, 72]. A rule of thumb for web sessions is to consider the session as concluded after 30 minutes of inactivity. A similar estimate does not exist for inactivity time between sessions, therefore we estimate the threshold t directly from log data.

Halfaker et al. [49] propose a data-driven approach to estimate the inactivity time for web sessions. We apply the same approach to estimate the task inactivity time between two consecutive sessions. We start by computing the inter-session times between 2 consecutive sessions from the same user. This is the difference between the starting time of the most recent session and the older session (see Equation (4.12)). Figure 4.9 shows the logarithmically scaled histogram of the inter-sessions times for all users in the dataset. We can observe a bi-modal distribution with a valley, similar to inter-action times for web users in [49]. We use Expectation Maximization (EM) to fit a two-component Gaussian mixture model. We assume that the distribution of inter-session times is a mixture of times corresponding to: (1) sessions belonging to the same task, i.e., the blue line in Figure 4.9, and (2) sessions belonging to different tasks, i.e., the orange line in Figure 4.9. The intersection point between the blue and orange lines represents the point where an inter-session time belongs to the two distributions with equal probability. We use this point, which corresponds to 10 days, as the threshold t. This means that 2 sessions belong to the same task if no more than 10 days elapsed between their starting times.

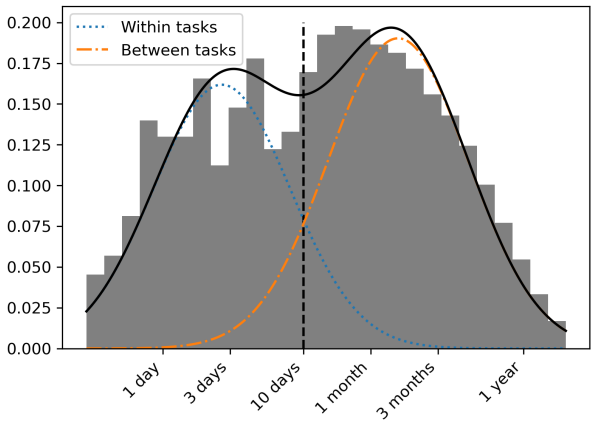


Figure 4.9: Histogram of logarithmically scaled inter-session times and fitted Gaussian mixture model.

4.4.3 Dataset Pre-processing

In the following, we describe all pre-processing steps. Tables 4.1 and 4.2, and Figure 4.8 are computed with the pre-processed dataset.

We removed items and actions that occur with very low frequency because they are not optimal for modeling. We remove all items with a frequency lower than 0.1%, and from actions, all sections, objects, and types with a frequency lower than 0.1%. Consecutive repeated actions of the same type, for example two clicks on the same object and section, are removed because they very likely represent noise due to the latency of the website or other connection issues. Very short sessions are poorly informative, thus we remove all sessions with less than 3 actions. On the other side, very long sessions lead to long training times, thus we truncate all sessions in the end to have a maximum of 30 actions (the 95th percentile). Based on the 10 days threshold estimated in Section 4.4.2, we discard all sessions with inter-session time greater than 10 days. Then, within the 10 days rule, we keep only the 7 most recent sessions for each user (the 95th percentile), which allows to reduce training times. We keep all users, even those with a single session and purchase.

4.5 Experiments

In this section, we outline the experimental set-up (Section 4.5.1) as well as present and analyse the results (Section 4.5.2). Our source code is publicly available[2].

[2]`https://github.com/simonebbruun/cross-sessions_RS/tree/main/extended`

4.5.1 Experimental Set-up

We start by describing the evaluation procedure, then the baseline models and finally our implementation and hyperparameter tuning.

Evaluation Procedure As a test set, we use the latest 10% of purchase events with associated past sessions. The remaining 90% is used for training. Since some users have had multiple purchase events, we remove purchase events from the training data, if their associated past sessions also appear in the test set. This resulted in 54 out of 48.381 purchase events in the training set being removed.

The models generate a score for how likely the user will buy each item, which is then sorted as a ranked list (i.e., the closer to the top of the ranking, the higher the estimated score of the item). Among the two types of items, new insurance products and additional coverages, it is only possible for a user to buy an additional coverage if the user has the corresponding base insurance product. For that reason, we use a post filter to set the score to the lowest score if that is not the case, as per Aggarwal [2]. The list of ranked items is evaluated with Hit Rate (HR), Mean Reciprocal Rank (MRR), Precision, Recall and Mean Average Precision (MAP). We use a cutoff threshold of 3 because: (1) the total number of items is 16, therefore high cut-offs (e.g., ≥ 10), will increase recall and all measures will reach high values, which will not inform on the actual quality of the RSs; (2) on the user interface the user will be recommended up to 3 items. Additionally, we report HR and MRR scores for all cut-offs value from 1 to 5.

Experimental results are supported by statistical testing. For HR we use McNemar's test [37] and for all other measures we use one-way ANOVA [81], both with a confidence level of 0.05, and post hoc tests to control the family-wise error rate due to multiple comparisons.

Baselines We compare our models against the following state-of-the-art baselines:

- **Random** recommends random items to the user.
- **Popular** recommends the items with the largest number of purchases across users.
- **SVD** is a method that factorizes the user-item matrix by singular value decomposition [35]. The portfolio data forms the user-item matrix, where a user-item entry is 1 if the user has bought the item and 0 otherwise. In the insurance domain it is likely for a user to buy the same item multiple times (e.g., a second car insurance), but matrix factorization cannot make repeated recommendations. Therefore, we add repeated items as new items (columns) to the matrix.
- **Demographic** is a classification model, as per Qazi et al. [111, 112], that uses user demographics and their portfolios as input features. The portfolios are represented with a feature for each item counting how many of the items the user has already bought. Demographic features and portfolio features are concatenated. We use a feed forward neural network to make a fair comparison with the neural session-based approaches.

- **GRU4REC** is a neural session-based model [59, 60] for single user sessions. As input, we use the last session of a user, consisting of the 3-tuple user actions described in Section 4.3.2. For every time step, the model outputs the likelihood for each action to be the one the user interacts with next. Recommendations are based on the output after the final time step.
- **GRU4REC Concat** is the same as GRU4REC, but all recent sessions of a user are concatenated into a single session.
- **SKNN_E** is a session-based nearest neighbour model as the one presented in Jannach and Ludewig [67] with the extension suggested in Latifi et al. [83]. The nearest neighbours are determined based on the set of actions in all recent sessions of each user. A user's set of actions is a vector computed with a maximum pooling operation over the actions generated by the user. We then adapt this baseline to our task, so the recommendations are based on the items purchased by the neighbours of the target user rather than the items interacted with in the ongoing session.
- **SKNN_EB** is the same as SKNN_E, but with a further extension suggested in Latifi et al. [83]: scores of items previously interacted with are boosted with a factor. The factor is tuned as a hyperparameter.

Note that the SVD and the demographic model make use of portfolio data, that is users' past purchases, while this is not the case for all cross-sessions models, GRU4REC and SKNN. GRU4REC is included as it has shown the best performance under identical conditions on various datasets among all the neural models compared in Ludewig et al. [97] and in Latifi et al. [83].

We tried the sequential extension to SKNN, Vector Multiplication SKNN, which is presented in Ludewig and Jannach [96], but did not obtain better results than the original one. Sequence and Time Aware Neighborhood [45] is not included as a baseline since it was not possible to adapt it to the task under consideration (for the reasons discussed in Section 4.2.2).

Implementation & Hyperparameters All implementation is in `Python 3.7.4` and `TensorFlow 2.6.0`[3]. We used `Adam` as the optimizer with TensorFlow's default settings for the learning rate, exponential decay rates and the epsilon parameter. Early stopping was used to choose the number of epochs based on the minimum loss on the validation set (explained below). We used two-layer networks[4] with dropout regularization on the first hidden layer.

We extracted a validation set from the training set in the same way as we extracted the test set from the whole dataset, so the validation set includes the latest 10% of purchases with associated sessions and the remaining is used for training. We tuned the hyperparameters of each neural model (batch size, number of units and dropout rate) on the validation set using grid search. We tested powers of 2 for the batch size and number of units ranging from 16 to 512.

[3]We used Tensorflow's implementation of padding and masking to deal with variable length input in the RNNs.

[4]In all models the second layer is a dense layer with ReLU activation function.

Table 4.3: Hyperparameters (*autoencoder/RNN)

Model	Batch size	Units	Dropout
Demographic	32	32	0.3
GRU4REC	32	256	0.2
GRU4REC Concat	32	256	0.2
Cross-sessions Encode	32	64	0.3
Cross-sessions Concat	128	64	0.3
Cross-sessions Auto*	128/32	512/64	-/0.4
Cross-sessions Encode with Weibull Loss	16	64	0.3
Cross-sessions Concat with Weibull Loss	128	128	0.3
Cross-sessions Auto with Weibull Loss*	128/64	512/128	-/0.4
Cross-sessions Encode with Attention	32	64	0.3
Cross-sessions Concat with Attention	128	64	0.3
Cross-sessions Auto with Attention*	128/32	512/64	-/0.4

For the dropout rate, we tested values in $[0.1, 0.5]$ with step size 0.1. The final hyperparameters used are reported in Table 4.3.

For GRU4REC and GRU4REC Concat we tried three different loss functions: cross-entropy, BPR [116] and TOP1 [59]. Cross-entropy was finally chosen for both models, as it performed best on the validation set. For the non-neural models, the optimal number of latent factors for the SVD model was 1, the optimal number of neighbours for both SKNN_E and SKNN_EB was 30, and the optimal boost factor for SKNN_EB was 0.5. Neural models were trained on Nvidia GeForce MX250 equipped with 2GB of GPU memory. The maximum training time was 6 hours.

4.5.2 Experimental Results

First, we compare our cross-sessions models against state-of-the-art baselines. Then we evaluate the best cross-sessions models combined with demographics. We further break down the performance of our models to understand the impact of the time of prediction, the number of sessions and actions, the order and recency of sessions as well as the size of the session threshold. Lastly, we conduct an ablation study to show how different actions (sections, objects and types) affect the performance of our models and a fairness analysis to show how the models perform on customers with different age, gender and income level.

Performance Analysis Table 4.4 presents a comparison of our cross-sessions models against the baselines. Unlike domains like retail and video services [59, 44], the simple popular model is quite a strong baseline, because of the small number of different items in the insurance dataset and the role of the post filter to make sure not to recommend items that the user cannot buy.

As seen in prior work on insurance RSs [111], we also see a significant improvement in using a demographic RS compared to the traditional matrix factorisation method, SVD. This is likely

Table 4.4: Performance results. All results marked with * are significantly different from Cross-sessions Encode with Attention. The best score for each measure is in bold. Percentages in brackets denote the difference between our models and the strongest baseline (SKNN_EB).

Model	HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Random	0.3235*	0.1114*	0.2940*	0.1910*	0.1839*
Popular	0.6217*	0.2145*	0.5855*	0.4764*	0.4540*
SVD	0.6646*	0.2372*	0.6327*	0.4997*	0.4829*
Demographic	0.7392*	0.2649*	0.7095*	0.5620*	0.5446*
GRU4REC	0.6479*	0.2313*	0.6208*	0.5443*	0.5264*
GRU4REC Concat	0.6616*	0.2365*	0.6362*	0.5620*	0.5453*
SKNN_E	0.8106*	0.2914*	0.7848*	0.6740*	0.6567*
SKNN_EB	0.8132*	0.2922*	0.7872*	0.6785*	0.6610*
Cross-sessions Encode	0.8380 (3.04%)	**0.3030 (3.67%)**	**0.8145 (3.46%)**	0.7093 (4.53%)	0.6923 (4.73%)
Cross-sessions Concat	0.8265 (1.62%)	0.2984 (2.12%)	0.8019 (1.87%)	0.7051 (3.92%)	0.6876 (4.02%)
Cross-sessions Auto	0.8356 (2.74%)	0.3024 (3.48%)	0.8128 (3.24%)	0.7085 (4.41%)	0.692 (4.69%)
Cross-sessions Encode with Weibull Loss	0.8378 (3.02%)	0.3013 (3.10%)	0.8120 (3.15%)	0.7039 (3.74%)	0.6852 (3.66%)
Cross-sessions Concat with Weibull Loss	0.8289 (1.92%)	0.2994 (2.44%)	0.8048 (2.23%)	0.7017 (3.41%)	0.6843 (3.53%)
Cross-sessions Auto with Weibull Loss	0.8352 (2.70%)	0.3031 (3.71%)	0.8124 (3.20%)	0.7076 (4.28%)	0.6909 (4.52%)
Cross-sessions Encode with Attention	**0.8385 (3.11%)**	0.3026 (3.56%)	0.8141 (3.41%)	**0.7118 (4.90%)**	**0.6941 (5.00%)**
Cross-sessions Concat with Attention	0.8317 (2.26%)	0.3016 (3.23%)	0.8090 (2.76%)	0.7091 (4.50%)	0.6928 (4.81%)
Cross-sessions Auto with Attention	0.8324 (2.36%)	0.3012 (3.08%)	0.8095 (2.83%)	0.7103 (4.69%)	0.6932 (4.87%)

Table 4.5: The versions of our cross-sessions encode models enhanced with demographic data. The rest of the notation is as in Table 4.4 (i.e., percentages in brackets denote the difference from the strongest baseline).

Model	HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode	**0.8542* (5.03%)**	**0.3103* (6.18%)**	**0.8313* (5.60%)**	0.7268* (7.11%)	0.7099* (7.41%)
Cross-sessions Encode with Weibull Loss	0.8529* (4.87%)	0.3100* (6.09%)	0.8310* (5.55%)	0.7219 (6.39%)	0.7055 (6.74%)
Cross-sessions Encode with Attention	0.8514* (4.69%)	0.3093* (5.83%)	0.8284* (5.23%)	**0.7301* (7.59%)**	**0.7134* (7.93%)**

due to the sparse feedback on items in the insurance domain (see Table 4.1) and the fact that users' demographic characteristics are good signals for learning insurance recommendations.

The session-based methods, SKNN_E, SKNN_EB and cross-sessions, significantly outperform the non-session-based methods, while this is not the case for GRU4REC and GRU4REC Concat. This shows that users' sessions are stronger signals for insurance recommendations than long-term preferences and demographic characteristics, but recommending the item that the user is most likely to interact with next on the website is not appropriate for insurance recommendations. Moreover, all the cross-sessions models significantly outperform SKNN_E and SKNN_EB suggesting that an RNN is better at modeling relationships between the user actions that lead to the purchase of specific items.

The results suggest that encoding of sessions is better than the trivial concatenation of sessions indicating that dependencies across sessions are important. The results further suggest that the encoding of sessions with a maximum pooling operation is better than an autoencoding of sessions. This is most likely because the order of actions (which the autoencoder takes into account) adds more noise to the model than signal, or the autoencoder needs a larger amount of training data in order to effectively learn to encode sessions. The best results of our cross-sessions

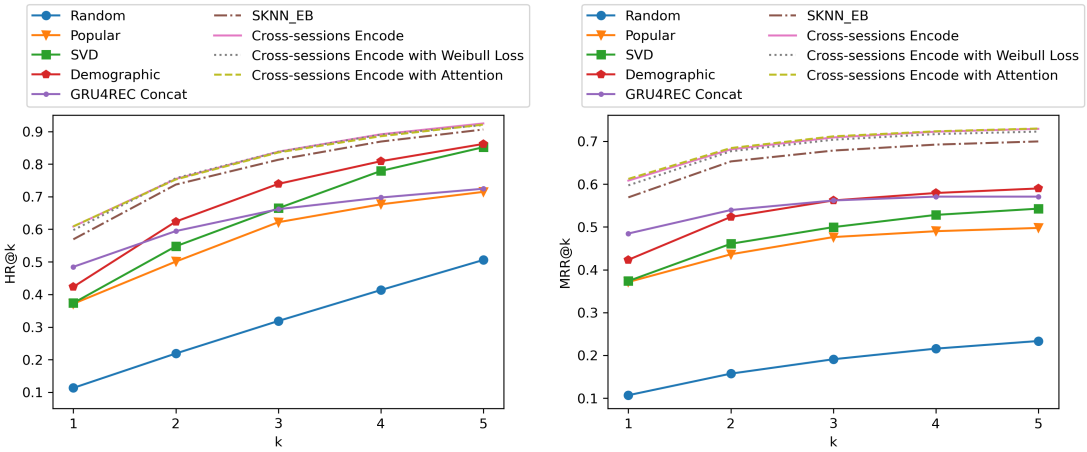


Figure 4.10: HR@k and MRR@k for varying choices of the cutoff threshold k.

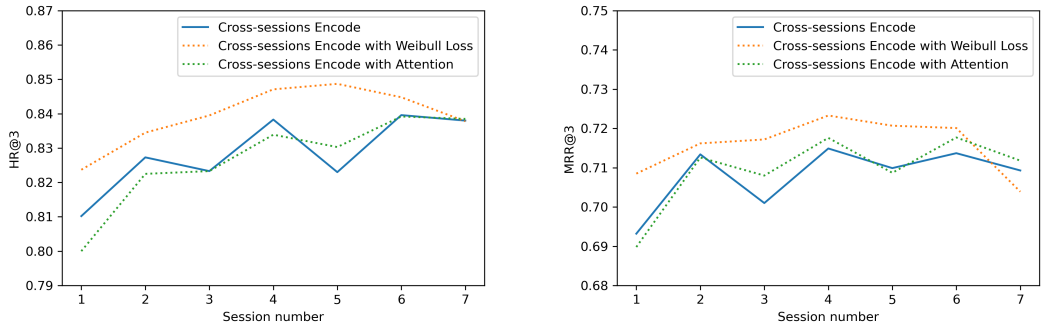


Figure 4.11: HR@3 and MRR@3 for each step in the users' sequences of sessions.

models trained with censored Weibull loss are obtained with Cross-sessions Encode and Cross-sessions Auto, but the models do not improve over Cross-sessions Encode trained with cross-entropy loss. Below, we further examine how the censored Weibull loss affects the performance at different times of recommendation. The performance of the cross-sessions models enhanced with attention mechanism slightly improves over the models without attention. The degree of improvement is similar across the three variations of encodings. Below, we further explore the attention mechanism by extracting the learned attention weights. Overall, encoding of sessions with a maximum pooling operation performed best. Hence in the rest of the analysis, we focus on Cross-sessions Encode trained with and without Weibull loss and attention mechanism.

Figure 4.10 shows HR and MRR at varying cutoffs k from 1 to 5. We observe similar results for recall, precision and MAP. The results are consistent over varying choices of k, with the exception of GRU4REC Concat which is better than SVD and Demographic for smaller cutoff thresholds (1-2), but not for larger. Across all choices of k there is a clear gap between the

cross-sessions models and the others. The general trend for both measures is that they tend to increase as the cut-off k increases. This is expected to happen since it is more likely to include the purchased items when k increases.

The results of our cross-sessions models combined with demographics are shown in Table 4.5. This hybrid approach yields better performance than the individual models for all evaluation measures. This indicates that the two types of information, sessions and demographic, capture different aspects of the problem. The best results are obtained with Cross-sessions Encode and Cross-sessions Encode with Attention and both models are significantly different from the best model without demographic data.

Overall, the results show that user sessions are stronger signals for insurance recommendations than insurance portfolios and users' demographics, but recommending the item that the user is most likely to interact with next on the website is not appropriate for insurance recommendations. Moreover, an RNN is better at modeling the relationship between user actions and purchases than an SKNN, and the results suggest that encoding of sessions is better than the trivial concatenation of sessions. Finally, the results show that the two types of information, user sessions and demographics, capture different aspects of the problem.

Analysis of the Time of Recommendation. Until now, we have evaluated all models after the final step in the users' sequences of sessions, when most information is available. However, when executing the models in a real-life scenario with online users, we do not know when each user will buy and we want to make recommendations already from their first session. In Figure 4.11 we compare our cross-sessions models, where we evaluate them for each step in the sequences of users' sessions. We see that the model trained with Weibull loss performs better than the models trained with cross-entropy loss for earlier steps. It is most likely because the censored Weibull loss takes into account the time of purchase. Hence, at the final step in the users' sequences of sessions, the two types of loss function result in similar performance, but the censored Weibull loss gives better results for earlier steps, which is desirable in a real-life scenario. We observe similar results for precision, recall and MAP.

Analysis of Number of Sessions and Actions Next, we analyse how the number of sessions affects our cross-sessions models. We break down the performance of our models based on the number of sessions, starting with only the most recent session, up to including all the available sessions (the maximum number of sessions per user is 7, see Sections 4.4.2 and 4.4.3). Figure 4.12 (left) shows HR@3 computed for our cross-sessions models for varying numbers of user sessions. In general, there is an increasing trend in performance with the number of recent sessions, emphasising the additional contribution brought by all recent sessions of each user rather than just the last one. We observe similar results for MRR, recall, precision and MAP.

We do the same analysis with the number of actions per session. Figure 4.12 (right) shows that HR@3 generally increases with the number of actions. The growth is particularly steep up to about 10 actions per session after which it flattens out. We observe similar results for MRR, recall, precision and MAP.

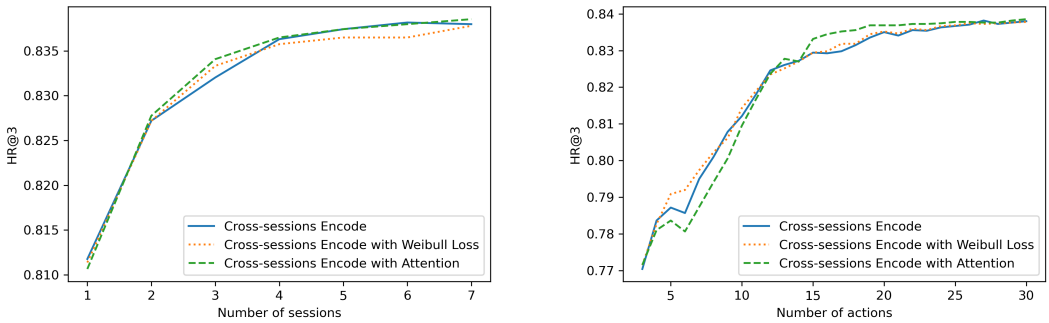


Figure 4.12: HR@3 for different number of sessions and actions.

Table 4.6: Study of session order. Relative change in parentheses.

Model		HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode	original session order	0.8380	0.3030	0.8145	0.7093	0.6923
	shuffled session order	0.8345 (-0.41%)	0.3008 (-0.7%)	0.8096 (-0.60%)	0.7058 (-0.49%)	0.688 (-0.61%)
Cross-sessions Encode with Weibull Loss	original session order	0.8378	0.3013	0.8120	0.7039	0.6852
	shuffled session order	0.8299 (-0.95%)	0.2992 (-0.68%)	0.8053 (-0.82%)	0.6991 (-0.68%)	0.6813 (-0.56%)
Cross-sessions Encode with Attention	original session order	0.8385	0.3026	0.8141	0.7118	0.6941
	shuffled session order	0.835 (-0.42%)	0.3012 (-0.46%)	0.8105 (-0.44%)	0.7038 (-1.12%)	0.6863 (-1.12%)

Analysis of Session Order We analyse the importance of session order by randomly shuffling the order of sessions, then retraining the models. We shuffle the order in both training, validation and test set, and perform the experiment 5 times to account for randomness. The mean performance is presented in Table 4.6. Across all our models and evaluation measures, performance drops when shuffling the session order, but the decrease is limited to less than 1.5%. The results indicate that the superiority of the cross-sessions models is not due to sequential dependencies, rather they are simply better at capturing the relationships between user actions and the purchase of specific items.

Analysis of Input Recency We analyse the importance of different time steps in the input sequence to our cross-sessions models, specifically, if more weights should be given to more recent input passed to the RNNs. We do this by extracting the attention weights, λ_i (see Equation (4.10)), from the cross-sessions encode model with attention mechanism. The weights are presented in Table 4.7. The model learns an attention weight for every step in the sequence of the user's past sessions. Since not all users have had 7 sessions (the maximum number of sessions), we present the attention weights averaged over users with the same number of sessions. For the users with only one session, the average attention weight is simply 1. For users with more than one session, we observe that the last session is assigned the greatest weight, while the previous sessions are weighted almost equally. This shows that the importance of input sessions does not decay linearly with the recency of the sessions, rather the last session is most important

Table 4.7: Attention weights averaged over users with the same number of sessions. Columns represent the number in the sequence of sessions and rows represent subsets of users with the same number of sessions.

Total number of sessions	Session number 1	2	3	4	5	6	7
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0.3059	0.6941
3	0	0	0	0	0.2326	0.2201	0.5473
4	0	0	0	0.1942	0.1735	0.1824	0.4498
5	0	0	0.1353	0.1509	0.1418	0.1537	0.4183
6	0	0.1453	0.1040	0.1169	0.1148	0.1350	0.3840
7	0.1243	0.1015	0.0867	0.1039	0.1112	0.1273	0.3451

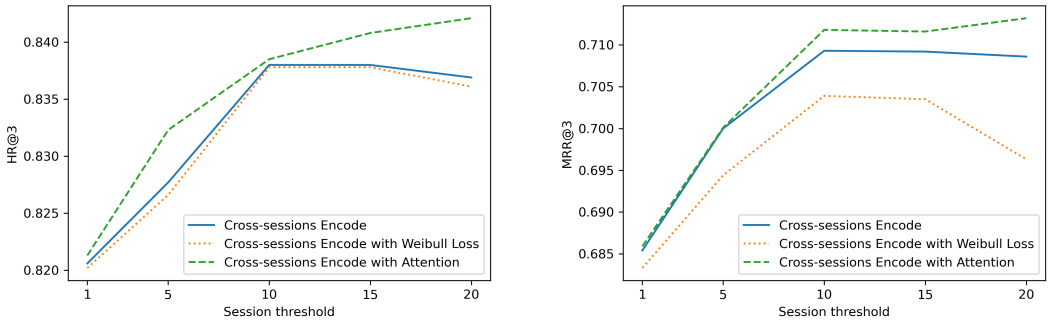


Figure 4.13: HR@k and MRR@k for varying choices of the session threshold.

and the rest of the sessions are equally important.

Analysis of Session Threshold In Section 4.4.2 we described how we estimated the 10 days threshold based on the approach in Halfaker et al. [49]. This threshold was then used in Section 4.4.3 to discard sessions that exceed 10 days. We now empirically analyse the estimated threshold by varying it, then retrain and evaluate how our models perform with different choices of threshold. The results are presented in Figure 4.13. We observe that the performance of our models drops when decreasing the threshold to less than 10, while an increase of the threshold has less impact on the performance. It shows that the additional sessions that are added to the input of our models when increasing the threshold above 10 are not used by the models. Only the model with attention has increasing performance when the threshold exceeds 10 days. It is likely due to the attention mechanism being able to use the right information from the additional sessions.

Table 4.8: Ablation study of actions. Relative change in parentheses.

Model		HR@3	Precision@3	Recall@3	MRR@3	MAP@3
Cross-sessions Encode	all actions	0.8380	0.3030	0.8145	0.7093	0.6923
	without E-commerce	0.7526 (-10.19%)	0.2698 (-10.95%)	0.7249 (-11.00%)	0.5951 (-16.09%)	0.5764 (-16.74%)
	without Claims reporting	0.8250 (-1.55%)	0.2979 (-1.68%)	0.8012 (-1.64%)	0.7006 (-1.22%)	0.6829 (-1.35%)
	without Information	0.8317 (-0.75%)	0.3000 (-0.98%)	0.8072 (-0.89%)	0.7045 (-0.68%)	0.6863 (-0.87%)
	without Personal account	0.8067 (-3.73%)	0.2894 (-4.48%)	0.7803 (-4.19%)	0.6604 (-6.89%)	0.6438 (-7.00%)
	without Items	0.7379 (-11.94%)	0.2652 (-12.46%)	0.7116 (-12.63%)	0.5720 (-19.36%)	0.5548 (-19.86%)
	without Services	0.8032 (-4.15%)	0.2880 (-4.93%)	0.7765 (-4.66%)	0.6639 (-6.40%)	0.6465 (-6.61%)
	without Start	0.8162 (-2.60%)	0.2935 (-3.11%)	0.7906 (-2.94%)	0.6771 (-4.55%)	0.6592 (-4.77%)
	without Act	0.8318 (-0.73%)	0.3005 (-0.80%)	0.8082 (-0.77%)	0.7035 (-0.82%)	0.6864 (-0.85%)
	without Complete	0.8317 (-0.75%)	0.3005 (-0.82%)	0.8078 (-0.82%)	0.7036 (-0.80%)	0.6861 (-0.89%)
Cross-sessions Encode with Weibull Loss	all actions	0.8378	0.3013	0.8120	0.7039	0.6852
	without E-commerce	0.7433 (-11.28%)	0.266 (-11.71%)	0.714 (-12.07%)	0.5791 (-17.74%)	0.5603 (-18.23%)
	without Claims reporting	0.8265 (-1.35%)	0.2971 (-1.40%)	0.8004 (-1.43%)	0.6901 (-1.97%)	0.6717 (-1.97%)
	without Information	0.8329 (-0.59%)	0.3006 (-0.22%)	0.8081 (-0.48%)	0.7013 (-0.37%)	0.6835 (-0.25%)
	without Personal account	0.8095 (-3.38%)	0.2887 (-4.19%)	0.7821 (-3.68%)	0.6674 (-5.18%)	0.65 (-5.14%)
	without Items	0.7349 (-12.28%)	0.2622 (-12.97%)	0.7053 (-13.14%)	0.5628 (-20.05%)	0.5446 (-20.52%)
	without Services	0.8096 (-3.37%)	0.2891 (-4.04%)	0.7817 (-3.73%)	0.6609 (-6.12%)	0.6429 (-6.17%)
	without Start	0.8149 (-2.73%)	0.2909 (-3.46%)	0.787 (-3.07%)	0.6722 (-4.50%)	0.6538 (-4.59%)
	without Act	0.8279 (-1.18%)	0.2979 (-1.11%)	0.8026 (-1.16%)	0.6993 (-0.65%)	0.6813 (-0.57%)
	without Complete	0.8283 (-1.13%)	0.2973 (-1.32%)	0.8019 (-1.24%)	0.6925 (-1.62%)	0.674 (-1.63%)
Cross-sessions Encode with Attention	all actions	0.8385	0.3026	0.8141	0.7118	0.6941
	without E-commerce	0.745 (-11.15%)	0.2675 (-11.61%)	0.7181 (-11.79%)	0.5938 (-16.57%)	0.5753 (-17.11%)
	without Claims reporting	0.8341 (-0.53%)	0.3002 (-0.82%)	0.8091 (-0.62%)	0.7036 (-1.15%)	0.6853 (-1.26%)
	without Information	0.8336 (-0.59%)	0.3015 (-0.36%)	0.81 (-0.50%)	0.7085 (-0.46%)	0.6914 (-0.38%)
	without Personal account	0.8147 (-2.84%)	0.2917 (-3.60%)	0.7886 (-3.13%)	0.676 (-5.02%)	0.6592 (-5.03%)
	without Items	0.7399 (-11.76%)	0.2645 (-12.61%)	0.7117 (-12.58%)	0.5696 (-19.98%)	0.5522 (-20.44%)
	without Services	0.8171 (-2.55%)	0.2927 (-3.27%)	0.7906 (-2.88%)	0.6741 (-5.29%)	0.6569 (-5.36%)
	without Start	0.8194 (-2.28%)	0.2953 (-2.44%)	0.7942 (-2.44%)	0.6831 (-4.03%)	0.6651 (-4.17%)
	without Act	0.8348 (-0.44%)	0.3002 (-0.82%)	0.8088 (-0.65%)	0.7034 (-1.18%)	0.6848 (-1.33%)
	without Complete	0.8385 (0%)	0.3028 (0.06%)	0.8142 (0.02%)	0.706 (-0.81%)	0.689 (-0.73%)

Analysis of Actions We use ablation to analyse the influence of different actions (i.e., sections, objects and types). Each time, we remove all actions of a given type and evaluate our cross-sessions models after retraining without the action type under analysis. The results are presented in Table 4.8 for all our models and evaluation measures. We did not consider the action type "click" in the ablation study because removing clicks results in removing most of the actions (65%), but also most of the objects and sections since users interact with objects and sections mainly through clicks.

For the Cross-sessions Encode and Cross-sessions Encode with Weibull Loss, all actions contribute positively to the model performance as their removal causes a drop in performance. We have a similar conclusion for the Cross-sessions Encode with Attention model with the exception that the removal of actions of the type "complete" increases performance instead of decreasing it, but the actual difference is negligible (less than 0.5%).

Not surprisingly, the most negative impact (up to -20.52% in MAP) is obtained when actions with the object type "item" are removed. Even if these are not the most frequent objects (see Table 4.2), they are highly informative as they provide information on the user's interests. The most frequent objects are "services", twice more frequent than items, and their removal affects negatively the performance even if with a less severe impact (up to -6.61% in MAP).

The second greatest negative impact is obtained when the actions from the section "e-

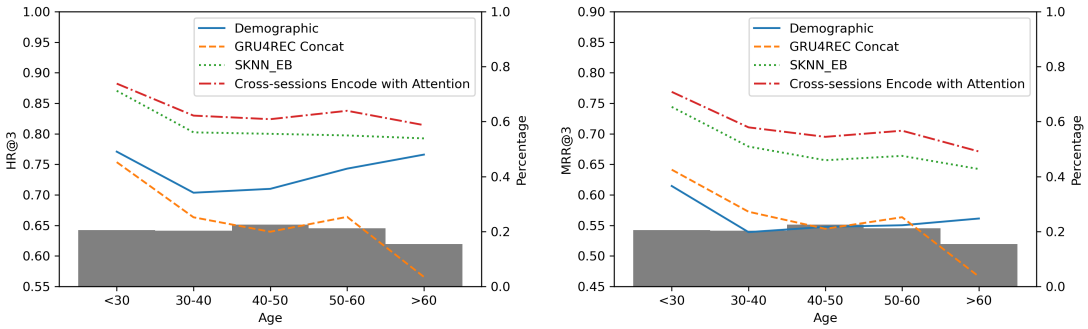


Figure 4.14: HR@k and MRR@k for different ages of the users. The grey bars illustrate the percentage of the different groups in the dataset.

commerce" are removed (up to −18.23% in MAP). Again, this is not the most frequent section, but it is highly informative since the user needs to access the e-commerce section to inspect and compare different insurance products. The most frequent section is the "personal account", which is 3 times more frequent than the e-commerce section. Its removal has a negative impact, but not as severe as for e-commerce (up to −7% in MAP).

In terms of action type, the removal of the type "start" has the greatest negative effect, even if limited with respect to the other two categories (up to −4.77% in MAP). The types "act" and "complete" have a negligible impact (less than 2%), but they also represent a very sparse signal (less than 5% of all types together).

Fairness Analysis We analyse group fairness of our models compared to the baseline models by exploring the performance for different age, gender and income level of the users, as these are protected characteristics that should not be discriminated against. The results of HR and MRR are illustrated in Figures 4.14, 4.15 and 4.16. The results for the remaining measures are similar, but not included for brevity. We further explore statistical significance, where Wald test is used for HR and t-test is used for MRR to compare the performance of the same model for different groups. In order to focus on variations between different types of models, we included the best-performing model among the non-sessions-based models, the best-performing version of the GRU4REC model, the best-performing version of the SKNN model and the best-performing model among our cross-sessions models.

We observe that the session-based models (GRU4REC Concat, SKNN_EB and Cross-sessions Encode with Attention) have lower performance when increasing age of the users. Especially the GRU4REC model has very poor performance for users from the age of 60, even though this group is well represented in the data. This is also supported by statistical tests, showing that the performance for users younger than 30 is significantly higher and the performance for users from the age of 60 is significantly lower than the rest. It does not apply to the same extent for the demographic model. It indicates that is it more difficult to learn insurance recommendations from user sessions generated by older users.

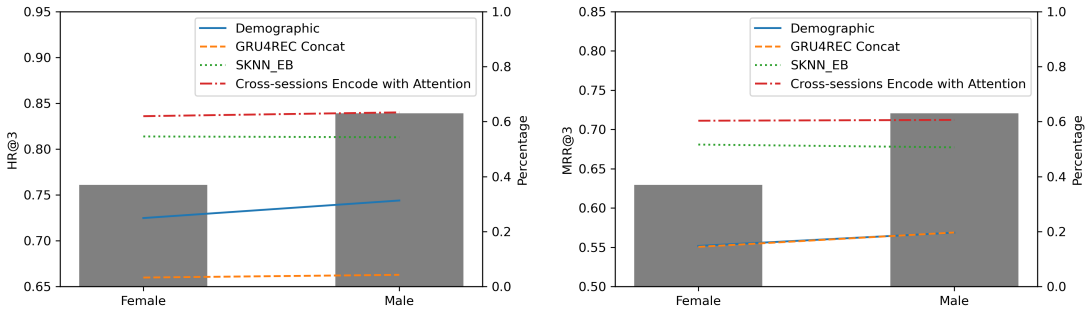


Figure 4.15: HR@k and MRR@k for different genders of the users. The grey bars illustrate the percentage of the different groups in the dataset.

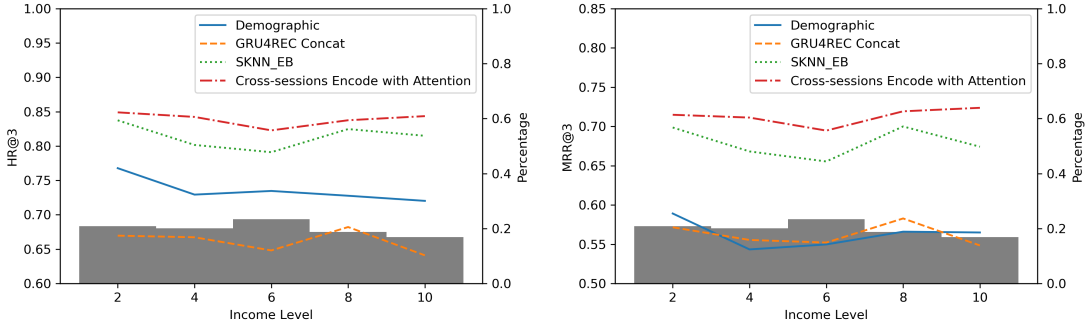


Figure 4.16: HR@k and MRR@k for different income levels of the users on a scale from $1-10$, where 1 is the lowest income level and 10 is the highest. The grey bars illustrate the percentage of the different groups in the dataset.

We further observe that none of the models are unfair towards gender, even though male users are over-represented in the dataset compared to female users. Only the demographic model has slightly better performance for male users than female users. Statistical tests show that none of the models have significantly different performances between genders.

Finally, we do not observe any strong trends in the performance for different income levels of the users. The demographic model has slightly decreasing performance for users with higher income level (only income level $1-2$ is statistically different from the others), while all the session-based models have slightly lower performance for users with middle-income level (it is only statistically significant for the SKNN model).

4.6 Conclusions and Future Work

The insurance domain is a data-scarce domain, where there is a limited amount of user feedback, interactions and number of items. These features make insurance recommendations particularly challenging. We have addressed the problem of insurance recommendation with our cross-sessions models, which learn to recommend items from past user sessions and different types of user actions. These actions are not always directly associated with items, for example a user reporting a claim on the insurance website. Unlike state-of-the-art session-based and session-aware models, our models predict a target action that does not occur within the session. Our cross-sessions models are all based on RNNs with GRU units. These are combined with 3 different ways to represent user sessions: maximum pooling encoding, concatenation of multiple sessions and an autoencoder approach; and 3 different loss functions and architectures: cross-entropy, censored Weibull and attention mechanism.

Experimental results on a real-world dataset show that all our cross-sessions models outperform several state-of-the-art baselines. Moreover, combining our models with demographic features boosts the performance even further. Additional analyses on the results shows that: (1) considering past user sessions is beneficial for cross-sessions models; (2) the removal of any type of action harms the performance, thus considering several types of actions is also beneficial. Finally, an analysis of group fairness with respect to age, gender, and income level, shows that our cross-session models are not biased against specific groups.

The output vector o, which was introduced in Section 4.3.2, contains a score for each item. The length of the vector is thereby equal to the total number of items. In item-poor domains like the insurance domain, the size of vector o is consequently small, but in other domains like music and retail, it is not rare to have hundreds of thousands of items. In such a case, calculating a score for each item would not be the most efficient approach. A possible solution could be to sample the output and only compute the score for a small subset of the items together with some negative examples. In future work, the effect of the size of vector o in our proposed models could be studied in large item domains with different sampling strategies.

As future work, we further plan to run an online A/B testing experiment to evaluate our models in an online scenario with users interacting in real-time. Moreover, we will investigate how to combine ML interpretability models [100], for example Local Surrogate (LIME), SHapley Additive exPlanations (SHAP), with our cross-sessions models to generate automatic explanations that will be shown to online customers. Finally, we will combine our models with other types of data, for example transcripts of user conversations over the phone, and develop a multi-modal extension of our models.

Chapter 5

Dataset and Models for Item Recommendation Using Multi-Modal User Interactions

Abstract

While recommender systems with multi-modal item representations (image, audio, and text), have been widely explored, learning recommendations from multi-modal user interactions (e.g., clicks and speech) remains an open problem. We study the case of multi-modal user interactions in a setting where users engage with a service provider through multiple channels (website and call center). In such cases, incomplete modalities naturally occur, since not all users interact through all the available channels. To address these challenges, we publish a real-world dataset that allows progress in this under-researched area. We further present and benchmark various methods for leveraging multi-modal user interactions for item recommendations, and propose a novel approach that specifically deals with missing modalities by mapping user interactions to a common feature space. Our analysis reveals important interactions between the different modalities and that a frequently occurring modality can enhance learning from a less frequent one.

5.1 Introduction

Recommender systems (RSs) most often learn from the users' actions such as ratings or purchases of items. In many domains, users interact with the system through multiple channels like website and call center, social tagging [40], image posting [80] and location sharing [10]. We refer to these different interaction types (e.g., clicks and speech) as multi-modal user interactions. We study the generation of recommendations when the user interactions have different modalities and thereby cannot simply be combined and used in existing recommender methods that are

designed for uni-modal user interactions. Previous work on multi-modal RSs exclusively focuses on multi-modal representations of items such as image, audio, and text [55, 152, 134, 91]. These methods are not designed for our scenario with multi-modal user interactions. In particular, existing methods only work when all modality information is available during training and inference. This is a problem when dealing with multi-modal user interactions where incomplete modalities naturally occur, since not all users interact through all the available channels. For example, in the insurance domain, where items are rather complex, some users purchase items through the website, while others prefer to make the purchase over the phone. Note that our scenario distinguishes from previous work on multi-behavior RSs [157, 160, 89], that try to infer user preferences with feedback of different categories, such as view, add-to-cart and purchase, but do not learn from user interactions of different modalities.

While multi-modal user interactions have great potential to be utilized in RSs, the lack of public datasets is a major roadblock to progress in this area. Therefore, the primary contribution of this paper is the creation and release of a real-world dataset to facilitate RSs research on multi-modal user interactions. The data is collected from a company dealing with insurance products for individuals and consists of (1) user sessions logged from the company's website, (2) transcribed conversations between users and the company's insurance agent, and (3) purchase actions. This data opens up novel research opportunities to predict product purchases based on rich, multi-modal interactions. Unlike commonly studied domains like movie, restaurant, or book recommendations, which are considered low-risk scenarios, the purchase of insurance products is a high-stakes domain where decisions can have a long-lasting impact on an individual's life, marking a significant departure from traditional RSs research. As our second contribution, we present and experimentally compare several approaches for combining different modalities for recommendation. We consider existing methods, such as different imputation approaches [26, 141, 144] and knowledge distillation [145]. We also propose a novel approach to jointly model different modalities, that suffer from naturally induced incompleteness, by mapping them into a common feature space. Specifically, we explore the following research questions.

RQ1 How can we effectively learn recommendations from multi-modal user interactions?

RQ2 How does it affect the quality of recommendations to jointly model the multiple modalities compared to separately?

A crucial factor for RQ1 is how we can represent multi-modal user interactions so they can be combined. In RQ2 it is essential to investigate if there are important interactions between the modalities and whether information from one modality can be useful when learning recommendations from another modality. Experimental results show that the two modalities represent different information that supplement each other well in the recommendation task. Compared to the existing methods, our proposed approach manages to capture important interactions between the modalities as well as use information from the most frequent modality when learning recommendations from the less frequent modality.

In summary, this paper makes the following contributions: (1) we create and release a dataset of multi-modal user interactions for the recommendation of financial products; (2) we

present and experimentally compare various approaches for leveraging multi-modal user interactions; and (3) we conduct an in-depth analysis of the results to shed light on what makes this problem challenging. All resources developed within this study, including the dataset and implementations of the models, are made publicly available on GitHub[1].

5.2 Related work

We present related work focusing on multi-modal recommendation datasets, RSs for the insurance domain, conversation-based recommendations, and multi-modal recommendation models.

Multi-modal Recommendation Datasets We contribute a novel dataset for multi-modal recommender tasks which is different from publicly available ones as follows. First, our dataset contains multi-modal user interactions while existing datasets contain multi-modal item representations, e.g., visual and textual representations of movies, books, and music [169], video and textual representations of micro-videos [139], and acoustic and textual representations of music [12]. Second, in our dataset missing modalities naturally occur since users might not interact through all channels, while in [169, 139, 12] missing modalities only occur due to technical reasons.

Insurance Domain In this domain, user feedback on items is sparse, because of few different items and because users rarely interact with insurance products. Most prior work supplements the small volume of user feedback with user demographics, such as age, income level, and employment. In this case, different techniques are used to categorize users based on demographic characteristics, then make recommendations within these categories [161, 122, 111]. Bi et al. [14] propose a cross-domain RS for the insurance domain. They use knowledge from an e-commerce source domain with daily necessities to learn better recommendations in the insurance target domain when data is sparse. In [20] and [22] a session-based approach is presented that uses a recurrent neural network (RNN) to learn insurance recommendations from web sessions with several types of user actions. None of the above methods use conversations or multiple modalities to learn insurance recommendations.

Conversation-based Recommendations The way we utilize conversations is not to be confused with conversational RSs, which are interactive systems that allow the user to disclose preferences, ask questions about items, and provide feedback [70]. Instead, we learn recommendations from past observed conversations and make recommendations in one-shot interactions. Few studies have focused on RSs based on past conversations. Gentile et al. [46] exploit e-mail conversations to learn user profiles utilizing different techniques (keyword extraction, extraction of named entities, and concept extraction). The profiles are then used to estimate the similarity between users. Rosa et al. [119] analyze text messages posted on social networks with the

[1] `https://anonymous.4open.science/r/RS_multi_modal_user_interactions-5893`

purpose of detecting users with potential psychological disorders (depression and stress). Then, if needed, an RS is used to send messages of happiness, calm, relaxation, or motivation. Text sentences are analyzed employing neural network approaches. Haratinezhad Torbati et al. [53] explore past online chats between users to learn search-based recommendations. They leverage language modeling techniques such as entity detection and entity-based expansions to represent the chats that are then used for ranking. However, none of these works combine the conversations with any other modalities.

Multi-modal Recommendation Models Multi-modal RSs can model relationships between different modalities and possibly benefit from the complementary and diverse sources of information that can not be captured by a uni-modal RS. He and McAuley [55] address cold start and data sparsity with matrix factorization on the user-item matrix complemented with visual representations. Wei et al. [152] use graph neural networks to learn the representation of each modality and then fuse them together as the final item representations. Instead of fusing the modality representations, Sun et al. [134] use knowledge graphs to include side information of different modalities. Liu et al. [91] introduce an attention neural network when fusing visual and textual representations, which can recognize users' varied preferences. All the above methods focus on multi-modal item representations, while we address the distinct challenge of multi-modal user interactions across different channels. Also, these methods rely on complete modality information during training and inference, which is unrealistic for multi-modal user interactions where not all users engage through all channels.

To deal with missing modalities, some imputation methods have been proposed in the broader field of machine learning. Tran et al. [141] concatenate all the modalities to form a large matrix, then apply a cascaded residual autoencoder to impute the missing elements. Cai et al. [26] use adversarial learning to complete the missing modalities, and Wang et al. [144] use a generative model to reconstruct the modality-specific embedding. In all these scenarios, the missing modalities existed in the real world but were missing in the dataset for technical reasons, such as problems with the measurement/tracking, or synthetically generated by removing observations from the dataset. This is a major difference from our scenario, where missing modalities are a result of "normal use." For instance, if a user did not interact with the website, then it might not be the most appropriate method to generate web sessions that did not take place. Furthermore, there might be some information in the user choosing not to have a conversation or web session, which is lost when using imputation. Wang et al. [145] use knowledge distillation to avoid imputation: they first train teacher models on each modality independently, then the student models are trained on complete modalities with soft labels from the teacher models and the true label. This method does not work well when only a small amount of the samples have complete modalities, as in our case.

5.3 Dataset

To the best of our knowledge, there exists no publicly available dataset that contains user interactions of different modalities and with a naturally induced incompleteness of modalities. We use a real-life dataset that we have obtained from a commercial insurance vendor and that we make freely available to the research community. Next, we describe the application context as well as the dataset.

5.3.1 Application Context

The data is collected from a vendor that deals with insurance for individuals. The items to be recommended are insurance products such as car, house and accident insurance or additional coverages, like roadside assistance for car insurance or chewing injury for accident insurance. The objective of an RS in this domain is to help customers continuously adjust their insurances to suit their needs.

The company's website is divided into four sections: (1) e-commerce, where users can buy insurance products, (2) claims reporting, where users can report insurance claims, (3) information section with information about payment methods, contact details, etc., and (4) personal account, where users can log in. In that way, users can interact with items, like buying a *car insurance* or reporting a *house insurance* claim, and can interact with services, like finding information about the *payment methods* or changing personal information such as *address*.

The company's call center has only inbound calls (i.e., the user calling the company). Here, the user can call for exactly the same purposes as on the website; for example, to buy insurances, report claims, ask about payment methods, and change address.

5.3.2 Dataset Description

The dataset was collected between May 1, 2022, to April 30, 2023. We collected *purchases* of insurance products and additional coverages made by existing customers. A purchase consists of one or more items bought by the same user at the same time. Both purchases made on the website and over the phone are included. For each user in the dataset, we collected all *conversations* that occurred before the user's purchase. A conversation consists of transcribed sentences from phone calls between a user and an insurance agent. The sentences in the dataset come with the order of the sentences, the speaker (user or agent), and the transcribed text. For each user in the dataset, we moreover collected all *web sessions* that occurred before the user's purchase. A web session consists of user actions on the insurance website. The actions in the dataset come with the order they are performed and action tags describing the section of the website in which the user interacts, the object on the website that a user chooses to interact with, and the way that a user interacts with objects.

All data has been anonymized to protect the identities and personal information of the individuals involved. Personal identifiers such as names, addresses, and contact information were removed; demographic information is not included. This ensures that individual participants

Table 5.1: Main properties of the dataset (*mean/std).

Users	51,877
Items	24
Purchases	62,401
Conversations	25,515
Web sessions	115,045
Conversations before purchase*	1.38/0.82
Web sessions before purchase*	2.3/1.98

cannot be identified or singled out from the dataset. The anonymization of data was carried out in compliance with applicable data protection laws and regulations. We thus expect this data to be useful for a long time without requiring frequent updates. We might release updated text embeddings as needed to accommodate the latest advancements in language models.

5.3.3 Dataset Pre-processing

The dataset is pre-processed in the following way. All items with frequency of less than 1% are removed from the purchases since low-frequency items are not optimal for modeling. Consecutive repeated actions of the same kind are discarded in web sessions because they very likely represent noise (e.g., double click due to latency). All conversations with less than four sentences and all web sessions with less than three actions are removed, as they offer limited insights. To avoid slow processing, all conversations and web sessions are truncated in the end to have a maximum of 541 sentences and 40 actions respectively (the 99th percentiles). Not all historical conversations and web sessions are kept for each user as only recent conversations and web sessions are assumed to be relevant to the current task. An inactivity threshold is used to define recent events (i.e., conversations/web sessions) such that two events belong to the same task if the time duration between them does not exceed a specific threshold, which is set to be 14 days based on [20]. Conversations and web sessions that exceed this threshold are thereby discarded. In addition within the 14 days rule, the list of recent events for each user is truncated to a maximum of 10 events (the 99th percentile) to reduce training time. Text embeddings are generated for each sentence in the conversations using a pre-trained language-specific BERT model [2] on the raw text. Keywords are extracted from the sentences by removing stop words, lemmatization, and using part-of-speech tagging to identify nouns.

Table 5.1 and Fig. 5.1 show general statistics of the dataset after pre-processing. There are approximately 62K purchases made by 52K different users. As observed from Fig. 5.1, not all users have had a conversation prior to their purchase (32%). Likewise, not all users have had a web session prior to their purchase (87%) why conversations and web sessions are naturally missing for part of the users. Only 19% of the users have had both conversations and web sessions prior to their purchase.

[2] `https://huggingface.co/Maltehb/danish-bert-botxo`

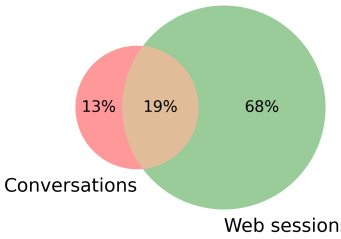


Figure 5.1: Distribution of the users having conversations and web sessions.

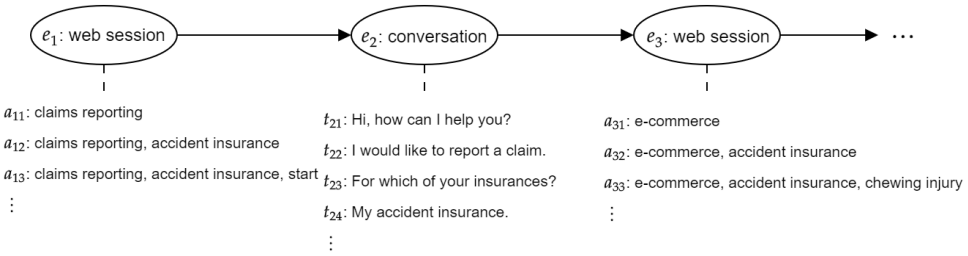


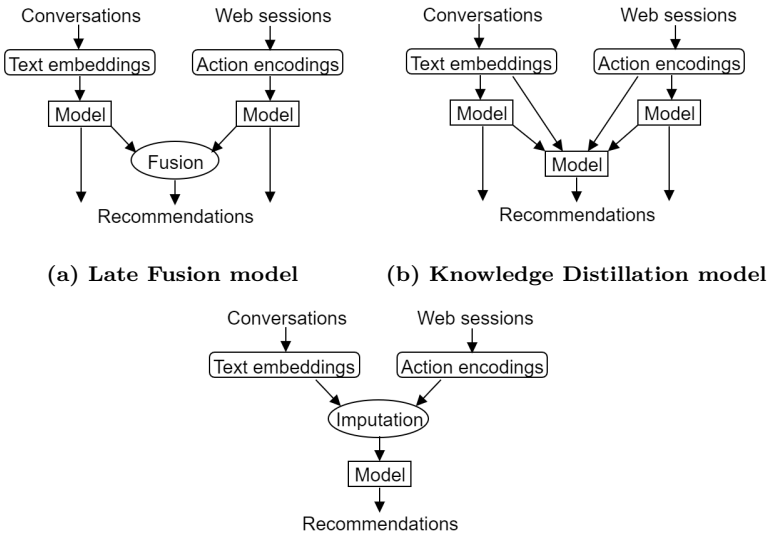
Figure 5.2: Example of a user's past events. An event can either be a *conversation*, in the form of text, or a *web session*, in the form of action tags.

5.4 Multi-modal Recommendations

The core modeling issue consists of representing multi-modal user interactions. Next, we formalize the problem for this new task (Section 5.4.1), present existing approaches (Section 5.4.2), and propose three models that specifically address the issue of naturally occurring incomplete modalities (Section 5.4.3). Together these models are meant to serve as a strong set of baselines for this new dataset.

5.4.1 Problem Formalization

The goal of our RS is to recommend the next items that a user will buy, given the user's past conversations and web sessions. As opposed to existing multi-modal RSs, which deal with items with multi-modal representations, in our task: (1) we deal with multi-modal user interactions, and (2) parts of the modalities are naturally missing. We collectively refer to web sessions and conversations as *events* and represent a user as the list of the user's past events, e_i, chronologically ordered. Depending on the user's history, the event list can either exclusively contain conversations, web sessions, or a combination of both. In the latter case, the order of conversations and web sessions can vary for different users. Moreover, the events have different modalities. A web session is a sequence of user actions, $\{a_{i1}, a_{i2}, a_{i3}, ..., a_{in}\}$, on the website, where an action is a set of tags. A conversation is a sequence of sentences, $\{t_{i1}, t_{i2}, t_{i3}, ..., t_{in}\}$, between a user and an insurance agent, in the form of text. Fig. 5.2 illustrates a user that has



(a) Late Fusion model **(b) Knowledge Distillation model**

(c) Imputation models

Figure 5.3: Schematic overview of the baseline models.

had a web session followed by a conversation, and finally a web session again.

The task is to learn a function, f, for predicting the probability that a user will buy each item j after the last event e_m based on the input sequence of a user's past events:

$$f(e_1, e_2, e_3, ..., e_m) = (\hat{p}_1, \hat{p}_2, \hat{p}_3, ..., \hat{p}_J), \tag{5.1}$$

where each element in $\{e_1, e_2, e_3, ..., e_m\}$ can either be a conversation or web session, $\hat{p}_j$ is the estimated probability that item j will be bought by the user, and J is the total number of items.

5.4.2 Existing Approaches

The following existing approaches can be applied to our problem.

- **Popular** recommends the items with the largest number of purchases across users.
- **Conversation** is a model trained only on the conversations. For that, we use a neural text classifier that takes as input the average text embeddings of a user's past conversations and predicts the purchase probability of each item. Note that this model only provides recommendations for users that had conversations.
- **Web Session** is a model trained only on the web sessions, using the session-based RS presented in [20], which takes into account the special characteristics of the insurance domain. Note that this model only provides recommendations for users that had web sessions.

- **Late Fusion** combines the output from the Conversation and Web Session models. It uses the recommendations from the Conversation model for those users that had only conversations and the recommendations from the Web Session model for those users that had only web sessions. Finally, it fuses the output from the two separate models for those users that have had both conversations and web sessions by averaging the predictions from the two models. This model is illustrated in Fig. 5.3a.
- **Knowledge Distillation** trains a joint model on users that had both conversations and web sessions. The model uses information from the two separate models as done in [145]. For users with only one modality, it uses the recommendations from the respective model for that modality; see Fig. 5.3b.
- **Generative Imputation** trains a model that generates the missing modality from the other modality as done in [26, 141, 144]. Once the missing modalities are imputed by the generative model, the modalities can be concatenated and jointly modeled. For a fair comparison, we use a neural network with the same architecture as the separate models. This model is illustrated in Fig. 5.3c.
- **Neutral Imputation** is similar to Generative Imputation. Because the modalities are not missing for technical reasons, we try a more neutral imputation strategy, so that fabricated conversations/web sessions do not corrupt the model. Missing conversations are imputed with the average text embedding in the training set and missing web sessions are imputed with the most frequent web session in the training set.

5.4.3 Proposed Methods

Next, we present novel models that aim to address the aspect of the problem concerning naturally induced incompleteness.

5.4.3.1 Data Representation

Our approach builds on top of the session-based RS proposed in Bruun et al. [20] that takes into account the special characteristics of the insurance domain, namely (1) web sessions consist of various actions, not only interactions with items; (2) users can have multiple web sessions before the target action (i.e., purchase); and (3) the purchase occurs outside the web session. We propose three different approaches to map the conversations into the same feature space as the web sessions and jointly model the two modalities with the session-based framework. In the session-based model, a user is represented by the sequence of the user's past web sessions $\{s_1, s_2, s_3, ..., s_m\}$. A web session can be encoded in different ways. Based on Bruun et al. [20], we encode a web session with a maximum pooling operation:

$$s_i = \max_{element}(a_{i1}, a_{i2}, a_{i3}, ..., a_{in}), \tag{5.2}$$

where $\max_{element}(\cdot)$ is a function that takes the element-wise maximum of vectors, and a_{ij} is a binarized vector of the action performed by a user. Then, the ordered sequence of encoded

sessions is passed through an RNN with gated recurrent units (GRU) that predicts what items the user will buy after the last time step. We extend this framework to the multi-modal case, such that a user is now represented by the sequence of the user's past events $\{e_1, e_2, e_3, ..., e_m\}$, where each event, e_i, can either be a conversation, c_i, or a web session, s_i. Next, we explain different ways of mapping the conversations and web sessions into the same feature space, so the events can be passed together through the RNN. In this way, a missing modality will not affect the model, since the events can be handled equally, once they are mapped into a common representation space.

5.4.3.2 Keyword Model

In the first way, called the *Keyword* model, the conversations are represented by keywords extracted from the text. We then manually match the keywords with the actions from the web sessions. Each event is thereby a binarized sequence of keywords, that can be encoded in the same way as the actions in Eq. (5.2), into a sequence of keyword vectors $\{k_1, k_2, k_3, ..., k_m\}$ representing the events. Then, for every time step i in the sequence of a user's events, an RNN with a single GRU layer computes the hidden state[3]

$$\begin{aligned} h_i &= (1 - z_i) \cdot h_{i-1} + z_i \cdot \hat{h}_i && \text{for } \; i = 1, .., m, \\ z_i &= \sigma(W_z k_i + U_z h_{i-1}), && (update \;\; gate) \\ \hat{h}_i &= \tanh(W k_i + U(r_i \cdot h_{i-1})), && (candidate \;\; gate) \\ r_i &= \sigma(W_r k_i + U_r h_{i-1}), && (reset \;\; gate) \end{aligned} \tag{5.3}$$

where W_z, U_z, W, U, W_r and U_r are weight matrices and $\sigma(\cdot)$ is the sigmoid function. The Keyword model is illustrated in Fig. 5.4a, with the corresponding neural architecture shown in Figure 5.5a.

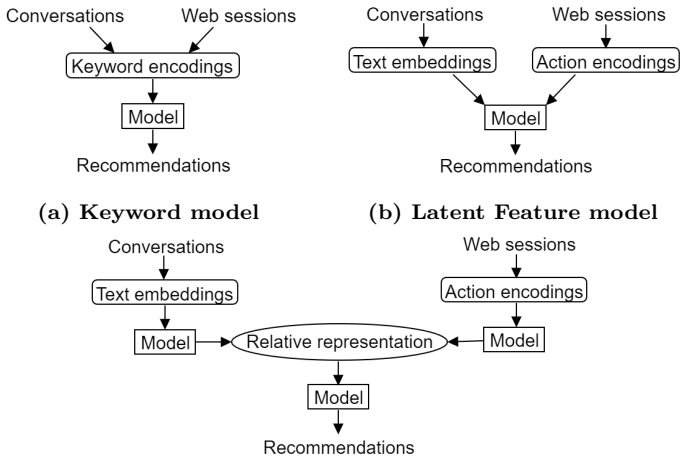
5.4.3.3 Latent Feature Model

The second way of mapping the two modalities into a common feature space is called the *Latent Feature* model. Here, conversations are represented by text embeddings, and web sessions are represented by action encodings (cf. Eq. (5.2)). The two different representations are passed as input to the same neural network, where the first layer maps them into a common representation of latent features. Formally, the following hidden state is computed for all the events in a user's sequence:

$$l_i = \tanh(W_c e_i \cdot \mathbb{1}_{\{e_i = c_i\}} + W_s e_i \cdot \mathbb{1}_{\{e_i = s_i\}}), \text{ for } i = 1, .., m, \tag{5.4}$$

where W_c and W_s are weight matrices used to map the conversations and web sessions respectively into the vector, l_i, of common latent features, and $\mathbb{1}_{\{\}}$ is an indicator function ensuring that W_c is used whenever the event is a conversation and W_s is used whenever the event is a session. We use the hyperbolic tangent as the activation function for this hidden layer to promote

[3]Note that bias terms are omitted when hidden states are presented.



(a) Keyword model **(b) Latent Feature model**

(c) Relative Representation model

Figure 5.4: Schematic overview of our models.

the task of mapping the real-valued conversations and the binarized sessions into features on the same scale. Then, for every time step in the sequence, l_i is passed through the hidden state in Eq. (5.3) instead of k_i. The weight matrices in Eq. (5.4) are optimized during the training of the full network. In that way, the latent features are automatically learned with respect to the task of generating recommendations. The Latent Feature model is illustrated in Fig. 5.4b, with the neural architecture shown in Fig. 5.5b.

In both cases, the neural network returns an output vector o of length J after the last time step. Because a user can buy multiple items at the same time, the learning task is considered a multi-label classification, and the sigmoid function is used on each element of o as the output activation function to compute the likelihood of purchase:

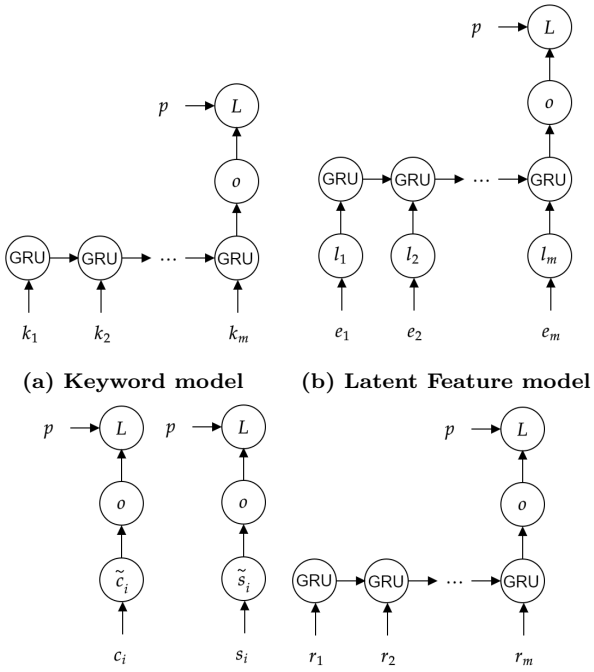
$$\hat{p}_j = \sigma(o_j), \quad \text{for} \quad j = 1, \ldots, J. \tag{5.5}$$

During training, the loss function is computed by comparing $\hat{p}$ with the binarized vector of the items purchased, p. Due to the learning task being multi-label classification, the loss function is defined as the sum of the binary cross-entropy loss over all items:

$$L = -\sum_{j=1}^{J} \Big(p_j \cdot \log(\hat{p}_j) + (1 - p_j) \cdot \log(1 - \hat{p}_j) \Big). \tag{5.6}$$

5.4.3.4 Relative Representation Model

We propose a third way, called the *Relative Representation* model. We use relative representation [101] that encodes the intrinsic information of a dataset learned by a neural network. Each data point becomes a set of coefficients that encode the point as a function of other data



(a) Keyword model **(b) Latent Feature model**

(c) Relative Representation model

Figure 5.5: Neural architectures of our models.

samples. It thereby enables the comparison of latent spaces across different neural networks. Hence, we train two separate neural networks from where latent representations of the two modalities can be extracted. We then use the method in [101] to make the two representations relative and thereby comparable to each other. The conversations and web sessions can now jointly be modeled by using their relative representations. Particularly, the two separate networks take individual conversations/web sessions as input (as opposed to the users' sequences of conversations/web sessions) and have an intermediate hidden state that can be used to create latent representations of single conversations/web sessions. They are trained on the same downstream recommendation task. Formally, a conversation, c_i, is represented by text embeddings and passed through a dense layer that computes a latent representation

$$\tilde{c}_i = \tanh(W_c c_i), \tag{5.7}$$

where W_c is a weight matrix. As in the Latent Feature model, we use the hyperbolic tangent as the activation function to promote the transformation of the different modalities into a common latent space. This representation is then used to solve the downstream recommendation task by passing it through another dense layer that computes the output o to be used in Eq. (5.5):

$$o = W_o \tilde{c}_i, \tag{5.8}$$

where W_o is another weight matrix. A web session, s_i, is represented by action encodings and similarly transformed into a latent representation, $\tilde{s}_i$, that can be used to solve the downstream

recommendation task:

$$\tilde{s}_i = \tanh(W_s s_i), \tag{5.9}$$

$$o = W_o \tilde{s}_i. \tag{5.10}$$

Once the weights are optimized with respect to the recommendation task, the first part of the networks (i.e., Eq. (5.7) and (5.9)) are used to convert conversations and web sessions into latent representations. Following the method in [101], a subset, $\mathcal{X}$, of the training set, denoted anchors, is selected. In our case, the anchors are selected among the users that are represented by both modalities. Given the indices of the anchor users in an arbitrary ordering $x_1, x_2, \ldots, x_{|\mathcal{X}|}$, the relative representation, r_i, of a conversation is computed as a function of the anchor users' conversations:

$$r_i = (sim(\tilde{c}_i, \tilde{c}_{x_1}), sim(\tilde{c}_i, \tilde{c}_{x_2}), \ldots, sim(\tilde{c}_i, \tilde{c}_{x_{|\mathcal{X}|}})), \tag{5.11}$$

where $sim()$ is a similarity function yielding a scalar score. Similarly, the relative representation of a session is given by

$$r_i = (sim(\tilde{s}_i, \tilde{s}_{x_1}), sim(\tilde{s}_i, \tilde{s}_{x_2}), \ldots, sim(\tilde{s}_i, \tilde{s}_{x_{|\mathcal{X}|}})). \tag{5.12}$$

A sequence of events is now represented by $\{r_1, r_2, \ldots, r_m\}$, where each r_i is either the relative representation of a conversation or a web session. The two modalities are jointly modeled by passing r_i through the hidden state in Eq. (5.3) instead of k_i for every time step in the sequence. This model is shown in Figs. 5.4c and 5.5c.

5.5 Experimental Setup

First, we describe the evaluation procedure, then the baselines, implementation details, and hyperparameter tuning.

5.5.1 Evaluation Procedure

As a test set, we use the latest 10% of purchases with associated past conversations and web sessions. The remaining 90% is used for training.

The models generate a score for how likely the user will buy each item, which is then sorted as a ranked list. There are two types of items: new insurance products and additional coverage. Since it is only possible for a user to buy additional coverage if the user has the corresponding base insurance product, we use a post filter to set the score to the lowest score if that is not the case, as per Aggarwal [2]. The list of ranked items is evaluated with Hit Rate (HR) and Mean Average Precision (MAP). Besides reporting HR and MAP averaged across all users in the test set (the union), we further break down the performance by the users that have only had conversations (the conversations-only), the users that have only had web sessions (the web sessions-only), and the users that have both had conversations and web sessions (the intersection), since this is of interest to our research questions. Because some of these subsets

Table 5.2: Hyperparameters

Model	Batch size	Units	Dropout
Conversation	512	64	0.2
Web Session	256	256	0.3
Knowledge Distillation	256	128	0.4
Generative Imputation	128	256	0.2
Neutral Imputation	128	128	0.2
Keyword	512	256	0.2
Latent Feature	512	256	0.3
Relative Representation	256	256	0.3

of users are relatively small (e.g., only 13% conversations-only), we report the average of the performance measures over five models trained from different seeds to account for randomness. We use a cutoff threshold of three because (1) the total number of items is 24, therefore high cut-offs (e.g., ≥ 10) will not inform on the actual quality of the RSs; (2) on the user interface the user will be recommended up to three items. Additionally, we report MAP scores for all cut-off values from one to five. Experimental results are supported by statistical testing. For HR we use McNemar's test [37] and for all other measures we use one-way ANOVA [81], both with a confidence level of 0.05.

5.5.2 Implementation & Hyperparameters

All implementation is in `Python 3.9.13` and `TensorFlow 2.10.0`. We used `Adam` as the optimizer with TensorFlow's default settings for the learning rate, exponential decay rates, and the epsilon parameter. Early stopping was used to choose the number of epochs based on the minimum loss on the validation dataset. We used two-layer networks[4] with dropout regularization on the first hidden layer. We partitioned the training set in the same way as the whole dataset, so the validation set includes the latest 10% of purchases with associated conversations/web sessions, and the remaining is used for training. Depending on the model, we remove action tags or keywords that have a frequency of less than 0.1 percent. We tuned the hyperparameters of each neural model (batch size, number of units, and dropout rate) on the validation set using grid search. We test powers of two for the batch size and number of units ranging from 64 to 256. For the dropout rate, we test values in $\{0.2, 0.3, 0.4\}$. The final hyperparameters used are reported in Table 5.2. In the Knowledge Distillation model, we use as distillation loss the sum of the binary cross-entropy loss over all items, as we are dealing with a multi-label classification task. The α and β parameters, which are used to control how much knowledge the student model gets from the teacher models, are set in proportion to how much data the student models are trained on. That is $\alpha = 0.32$ for the Conversation model and $\beta = 0.87$ for the Web Session model. In the Relative Representation model, we use cosine similarity as the similarity measure (see Eqs. (5.11) and (5.12)). We tune the number of anchors on the validation set and find the optimal number to be 125 which are sampled from each class in the training set.

[4]In all models the second layer is a dense layer with ReLU activation function.

Table 5.3: Performance results. All results marked with * are significantly different from the Latent Feature model. The best scores for each measure are boldfaced. Percentages in brackets denote relative differences w.r.t. the strongest baseline (Imputation).

Model	Union		Conversations-only		Web Sessions-only		Intersection	
	HR@3	MAP@3	HR@3	MAP@3	HR@3	MAP@3	HR@3	MAP@3
Popular	0.4595*	0.2742*	0.5249*	0.3318*	0.4287*	0.2424*	0.5111*	0.3340*
Conversation	-	-	0.6623*	0.4975	-	-	0.6184*	0.4497*
Web Session	-	-	-	-	0.6642*	0.5188	0.6512*	0.4750*
Late Fusion	0.6703*	0.5179*	0.6623*	0.4975	0.6642*	0.5188	0.6949*	0.5287
Knowledge Distillation	0.6697*	0.5168*	0.6623*	0.4975	0.6642*	0.5188	0.6921*	0.5235*
Generative Imputation	0.6685*	0.5156*	0.6582*	0.4878*	0.6635*	0.5180	0.6910*	0.5270
Neutral Imputation	0.6767*	0.5161*	0.6843*	0.5053	0.6687	0.5151	0.6966*	0.5265*
Keyword	0.6840 (1.08%)	0.5250 (1.74%)	0.6940 (1.42%)	0.5198 (2.86%)	0.6681* (-0.09%)	0.5141 (-0.18%)	**0.7270 (4.37%)**	**0.5629 (6.91%)**
Latent Feature	**0.6872 (1.55%)**	**0.5339 (3.46%)**	**0.7145 (4.42%)**	**0.5415 (7.17%)**	**0.6712 (0.38%)**	**0.5238 (1.69%)**	0.7183 (3.12%)	0.5603 (6.44%)
Relative Representation	0.6846 (1.16%)	0.5308 (2.85%)	0.7105 (3.84%)	0.5369 (6.25%)	0.6696 (0.14%)	0.5216 (1.27%)	0.7138 (2.46%)	0.5554 (5.49%)

5.6 Results

Next, we compare the different approaches presented in Section 5.4.3 in order to answer our research questions. Table 5.3 presents performance results. Figure 5.6 shows MAP at varying cutoffs k. We have similar results for HR which are omitted to save space.

RQ1 How can we effectively learn recommendations from multi-modal user interactions?

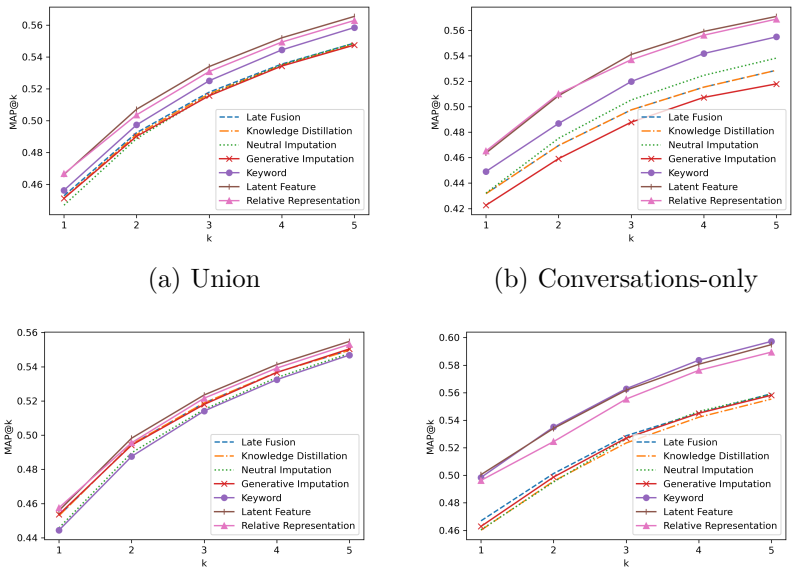
We look at the overall results of the models on the union (i.e., the entire dataset). The results show that even though there are few items, all the models outperform the simple Popular baseline considerably, showing there is information in the multiple modalities that can be learned.

Simply averaging the predictions from the two separate models, as is done in the Late Fusion model, proves to be a strong baseline; we observe that the Knowledge Distillation and Imputation models do not manage to improve over the Late Fusion model by jointly modeling the two modalities. It is reasonable that the training data is too small for the Knowledge Distillation model which is trained on the intersection data alone, even though it uses information from the two separate models. For the Imputation models, especially the Generative Imputation model, it is likely that the synthetic imputation adds too much noise to the data.

The results show that our approach of mapping the two modalities into the same feature space significantly outperforms all the existing methods with the Latent Feature model being the best.

RQ2 How does it affect the quality of recommendations to jointly model the multiple modalities compared to separately?

We look further into the results broken down by conversations-only, web sessions-only, and intersection. We observe that the Web Session model performs better than the Conversation model on the intersection. This is likely because there are considerably more web sessions than conversations in the training data. Note that the Late Fusion and the Knowledge Distillation models use the recommendations from the separate models on the conversations-only and web sessions-only, hence the identical performance in these cases.



(a) Union

(b) Conversations-only

(c) Web Sessions-only

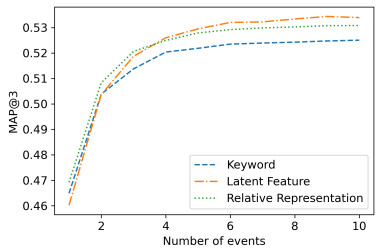
(d) Intersection

Figure 5.6: MAP@k for varying choices of the cutoff threshold k.

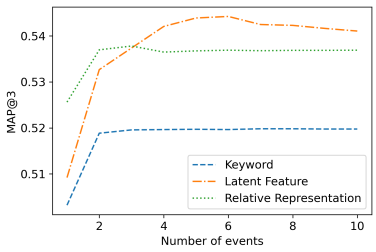
We find that the strength of the Late Fusion model is due to a considerable improvement on the intersection compared to the Conversation and Web Session models. This shows that the modalities capture different aspects of the problem. The results further show that the Keyword, Latent Feature and Relative Representation models successfully manage to model important interactions between the two modalities, while this is not the case for the Knowledge Distillation and Imputation models, as performance increases considerably on the intersection compared to the baselines, with the Keyword model being best.

The Neutral Imputation model improves slightly on the conversations-only subset, suggesting that the smaller conversation data benefits from the larger web session data when jointly modeling the two modalities. This does not apply to the Generative Imputation model, even though the modalities are also jointly modeled with this model. We observe that the Keyword, Latent Feature and Relative Representation models all improve considerably on the conversations-only subset while preserving the same performance on the web sessions-only subset compared to the existing methods. It shows that the small conversation dataset successfully benefits from the larger web session data when mapping the two modalities into the same feature space. The improvement is greatest with the Latent Feature and Relative Representation models. It is likely due to a loss of information when representing the conversations by keywords compared to text embeddings.

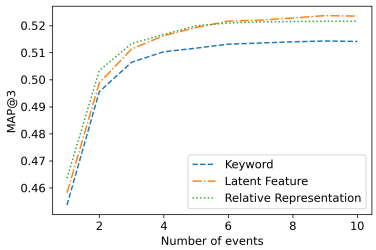
Figure 5.6 shows that the results are consistent across various cutoff thresholds. Across all cutoff values, there is a clear gap between our proposed models and the existing methods with the exception of the Keyword model on the web sessions-only where performance does not improve over the existing methods.



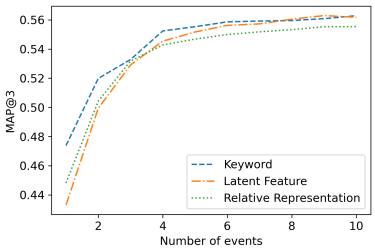
(a) Union



(b) Conversations-only



(c) Web Sessions-only



(d) Intersection

Figure 5.7: MAP@3 for a different number of events.

Note that our models do not have more steps, more parameters or longer training times than the existing models, why they are not more computationally expensive.

5.7 Analysis

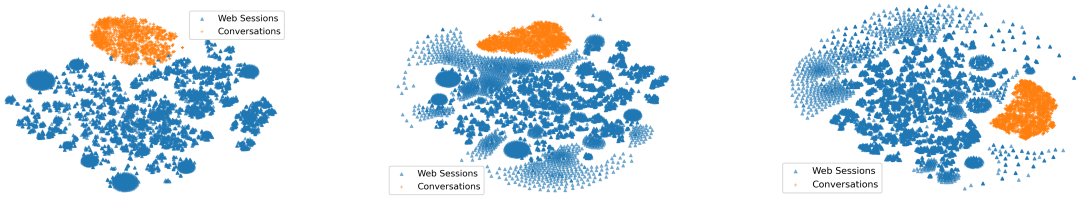
We conduct further analysis to understand what makes this dataset and problem challenging. We break down the performance of our proposed models to understand the impact of the number of past conversations and web sessions as well as the order of them. Then we use t-SNE to visualize the representations of the modalities before and after they are passed through the neural models.

5.7.1 Number of Events

We analyze how the number of events (i.e., conversations and web sessions) affects our models. Figure 5.7 shows MAP@3 broken down by the number of events, starting with only the most recent event, up to including all the available events. In general, performance increases with the number of events up to about five events, after which it flattens out. It is less important for users with only conversations to include many historical events. Particularly, the Keyword and Relative Representation models do not benefit from more than two/three conversations. The Relative Representation model generally outperforms the two others when only a few events are included, except for the intersection where the Keyword model is best. It is likely because the relative representations are computed individually for each conversation and web session,

Table 5.4: Study of the event order. Relative changes are in parentheses.

Model		Union		Conversations-only		Sessions-only		Intersection	
		HR@3	MAP@3	HR@3	MAP@3	HR@3	MAP@3	HR@3	MAP@3
Keyword	original event order	0.6840	0.5250	0.6940	0.5198	0.6681	0.5141	0.7270	0.5629
	shuffled event order	0.6775 (-0.95%)	0.5216 (-0.65%)	0.6935 (-0.07%)	0.5181 (-0.34%)	0.6645 (-0.54%)	0.5154 (0.26%)	0.7073 (-2.71%)	0.5433 (-3.49%)
Latent Feature	original event order	0.6872	0.5339	0.7145	0.5415	0.6712	0.5238	0.7183	0.5603
	shuffled event order	0.6722 (-2.18%)	0.5164 (-3.27%)	0.7135 (-0.14%)	0.5300 (-2.13%)	0.6610 (-1.52%)	0.5133 (-2.01%)	0.6791 (-5.45%)	0.5171 (-7.71%)
Relative Representation	original event order	0.6846	0.5308	0.7105	0.5369	0.6696	0.5216	0.7138	0.5554
	shuffled event order	0.6776 (-1.03%)	0.5232 (-1.43%)	0.6992 (-1.59%)	0.5314 (-1.03%)	0.6664 (-0.49%)	0.5183 (-0.63%)	0.6978 (-2.23%)	0.5329 (-4.05%)



(a) Keyword model (b) Latent Feature model (c) Relative Representation model

Figure 5.8: t-SNE visualization of the input representations.

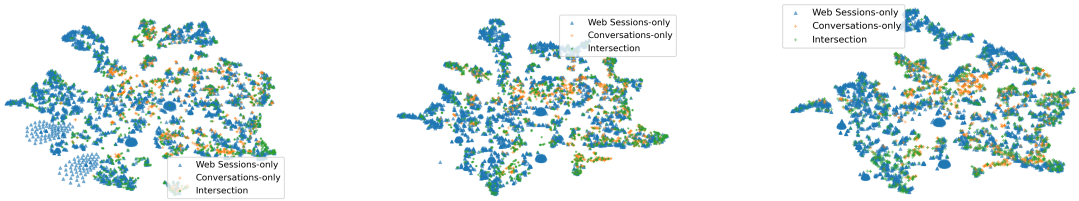
making it less dependent on the whole user history. We observe similar results for HR, which are not included due to space constraints.

5.7.2 Event Order

We analyze the importance of event order by randomly shuffling the order of events and retraining the models. We shuffle the order in both training and test data and perform the experiment five times to account for randomness. The mean performance is presented in Table 5.4. We observe that the performance of the Keyword model drops less than that of the Latent Feature and Relative Representation models on the union (less than -0.95%). It shows that the Latent Feature and Relative Representation models are more successful in capturing dependencies in the sequential order of events. In addition, it is also likely that these models overfit the data, as they have more parameters than the Keyword model. The Latent Feature model is more affected than the Relative Representation model when shuffling the order. It is likely because the relative representations are computed individually for each conversation and web session while the latent features are learned from the users' sequences of conversations/web sessions. For all models, the biggest drop in performance is seen on the intersection (up to -7.71%). It explains part of the superiority of mapping the modalities into a common feature space since sequential dependencies across the two modalities prove to be important.

5.7.3 Visualization

We use t-SNE to visualize the input representations of our three different approaches in Fig. 5.8. That is, we visualize the keywords, the latent features, and the relative representations, respectively. We observe that the conversations and web sessions are clustered together, showing that



(a) Keyword model (b) Latent Feature model (c) Relative Representation model

Figure 5.9: t-SNE visualization of the model output.

the two modalities contain different information about the users. It likely differs from RSs for items with multi-modal representations, where for instance an image and a text description of the same item typically contain more similar information about the item. In Fig. 5.9, we visualize the output after the conversations and web sessions have been passed through the neural models. Now modalities of the same type are more separated, as the different information is learned as signals for the same recommendations. Note that the users with both conversations and web sessions (the intersection) have one joint output. Overall, the visualization shows that the gain of learning recommendations from multi-modal user interactions is partly because the modalities contain different information about users that complement each other well for the task.

5.8 Conclusion and Future Work

We have taken an important first step in the problem of learning recommendations from multi-modal user interactions. This is a highly relevant problem to which no satisfying solution currently exists, as prior work and public datasets focused only on items with multi-modal representations and complete modalities. Our contributed dataset contains real-world user interactions of multiple modalities in terms of website actions and call center conversations that are naturally missing for some users. Experimental comparison of several approaches for combining the modalities reveals that they contain very different information that complement each other well for the recommendation task. Investigation of three new ways of representing the modalities shows that a model which automatically learns latent features is most effective while a model based on relative representations has the advantage of being less dependent on long user history. Finally, a method using keyword extraction is particularly good at capturing feature interactions between the modalities. Overall, we demonstrate that it is particularly beneficial to include user interactions of different modalities for generating effective personalized recommendations.

Since this work focuses on the fusion part of different modalities, the conversations are treated as generic text and could be any text related to a user such as e-mails, chats, and social media posts. As future work, we plan to do a dedicated work on effectively learning recommendations from past conversations by taking into account the context, like time and speaker.

Chapter 6

Feature Attribution Explanations of Session-based Recommendations

Abstract

Session-based recommender systems often employ black-box models to dynamically make recommendations based on current session interactions. However, explaining why an item is recommended is needed, not only to increase the trust and transparency of the system but also as a requirement under various legislations. One popular approach to generate explanations of recommendations is *additive feature attribution*, which assigns a score of how much each feature contributed to the prediction, such that the sum of the scores equals the prediction score. In this work, we posit that applying additive feature attribution to explain session-based recommendations suffers from two major limitations. First, additive feature attribution does not reflect the attribution coming from **sequential dependencies** of session interactions learned by the recommendation model. It assumes that interactions occur independently of each other. Second, as additive feature attribution relies on independent features, it fails when the features are correlated due to **repeated interactions** in sessions. We empirically verify the impact of these limitations upon explanation faithfulness. We further rectify these limitations, by presenting a simple occlusion-based feature attribution approach that is specifically tailored to session-based recommendations. Our method computes joint feature attributions for sets of interactions with sequential dependencies and sets of repeated interactions to account for their non-linear relations. Experimental results on multiple datasets and recommendation models confirm the superiority of our proposed method over state-of-the-art attribution-based explanation methods.

6.1 Introduction

Recommender systems (RSs) often make use of complex models that are not easily explainable. Providing explanations for why an item is recommended to a user can increase the trust and

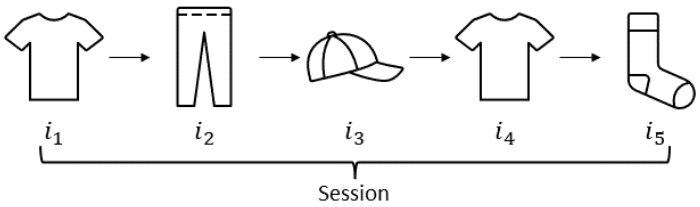


Figure 6.1: Example of a session in the e-commerce domain. A user interacts with items in a sequential order and can interact with the same item multiple times. Here interaction i_1 and i_4 are with the same item.

transparency of an RS. Moreover, recent legislation makes it compulsory to offer recommendations that are explainable by humans.

Explanations can either be global and explain the entire model behavior or local and explain an individual prediction. This work concerns local explanations that explain the recommendations given to a single user. Explainability methods can further be divided into intrinsic and post hoc methods. With an intrinsic method, explainability is achieved by restricting the complexity of the model, while a post hoc method achieves explainability by applying methods that analyze the model after training. Post hoc explainability methods can be used to explain most RSs as they can operate over a variety of trained models, while not affecting predictive performance. A major line of work within post hoc explainability methods is additive feature attribution methods, such as local interpretable model-agnostic explanations (LIME) [117], Shapley additive explanations (SHAP) [98] and integrated gradients (IG) [138]. These methods explain a model prediction by assigning attribution scores to individual input features, such that the sum of the scores equals the prediction score. The feature attributions indicate how much each feature in a model contributed to the predictions for each given instance. Additive feature attribution methods are popular because they are easy to implement and intuitive to humans. However, we posit that these methods are not suitable for explaining session-based RSs that dynamically provide recommendations based on a user's interactions in an ongoing session, instead of user profiles based on static preferences. In this paper, we identify the reason why additive feature attribution fails due to two special characteristics of session-based RSs namely (1) sequential dependencies, occurring when the recommendation model learns relationships in the order in which a user interacts with items; and (2) repeated user interactions occurring when a user interacts with the same item multiple times in a session (see Figure 6.1 for an example of a session).

Previous work only covers intrinsic explainability methods for session-based RSs, which affect the recommendation accuracy [30, 168, 153], or the application of additive feature attribution to the traditional matrix completion task in RSs [103, 19, 118], which do not deal with the issues of sequential dependencies and repeated interactions that are present in session-based RSs. To our knowledge, no prior work has addressed the two problems of additive feature attribution for session-based RSs that we identify.

In this paper, we empirically confirm the extent of these issues for session-based RSs. In

addition, we present a feature attribution approach that is designed to overcome these limitations in the setup of session-based RSs. To the best of our knowledge, both of the above are novel contributions.

6.2 Related Work

In this section, we first overview related work that studies different methods particularly designed to explain session-based RSs. Then we present related work that focuses on additive feature attribution for explaining recommendations.

6.2.1 Explaining Session-based Recommender Systems

Previous work on this topic includes studies on session-based RSs that are inherently explainable (i.e. intrinsic explainability methods). Chen et al. [30] propose a session-based RS based on data mining to make it inherently explainable. Their model generates recommendations and explanations by considering three factors: sequential patterns, repetition clicks and item similarity. First, candidate items are selected according to users' sequential patterns and repeated interactions, and then they are ranked by considering short-term interest with item similarity computations. Zheng et al. [168] use a knowledge graph with item attributes to explore item dependencies in sessions that can be used to generate recommendations. Then they explore pathways over the knowledge graph that can explain the recommendations. Wu et al. [153] explore the relationship between items from a causality and correlation perspective to construct graphs from sessions that are then modeled with a graph neural network recommendation approach. They use the separation between causality and correlation to facilitate the explanation of recommendations. All the above methods of explaining session-based recommendations are model-specific and can thereby not be used to explain any session-based recommendation model and explainability is achieved by restricting the complexity of the model which will affect the recommendation accuracy.

6.2.2 Additive Feature Attribution of Recommender Systems

Other work has studied additive feature attribution methods for generating post hoc explanations of RSs that are not model-specific. Most prior work extends or adapts the simplification-based method LIME [117] or the perturbation-based method SHAP [98], two well-known feature attribution methods. For each instance in the dataset, LIME trains a linear model to approximate the local decision boundary for that instance. SHAP is based on the Shapley Values approach that computes the average marginal contribution of each feature across all possible feature perturbations. Nóbrega and Marinho [103] propose an adaption of LIME to explain traditional RSs that model long-term user profiles. They use real data instead of synthetic ones (as normally used in LIME) for training the interpretable model. The idea is to fix the user of interest and sample items according to their empirical distribution. Brunot et al. [19] introduce

another approach of sampling around the recommendation instance in LIME. As opposed to [103], they use an out-of-sample prediction mechanism to predict ratings for yet unseen perturbed users instead of limiting the perturbation to existing users. Moreover, they show that it improves the quality of explanations to consider implicit preferences as expressed by comparisons of pairs of ratings instead of considering the ratings directly. Roberts et al. [118] evaluate LIME and SHAP on a movie recommendation task. They find that LIME provides the most faithful explanations in dense data segments while SHAP is the best in sparse segments. They trace the superiority of SHAP in sparse segments to the completeness constraint property inherent in SHAP and introduce the same constraint into LIME to obtain an explanation method that performs well on both dense and sparse data. All the above studies only apply to the traditional matrix completion task, and no previous work has evaluated or adapted additive feature attribution for the session-based recommendation task.

6.3 Limitations of Additive Feature Attribution

6.3.1 Problem Definition

We consider a *session* as consisting of a user's *interactions* with items, for example clicks or views. Formally, we define a user's session, s, with interactions $\{i_1, i_2, ..., i_n\}$ and a session-based recommendation model $f(\cdot)$ that generates a recommendation score from the input session. In this paper, we focus on local explanations within sessions. That is, we are interested in explaining the recommendations given to a single user (in a single session). Specifically, we focus on explanations in the form of feature attribution scores that reflect how each feature contributed to the prediction[1]. In the framework of session-based RSs, the features are the user's interactions with items. Hence, the explanation of $f(s)$ constitutes an attribution score ϕ_k for each interaction, i_k, in the input session. The problem to be solved here is how to compute the best attribution score ϕ_k for each user interaction in a session.

6.3.2 Limitations

The above problem has been currently addressed by additive feature attribution methods, such as LIME, SHAP and IG. These methods compute feature attributions, such that the sum of the attribution scores equals the prediction, as explained in Section 6.2. We posit that these methods are not suitable for explaining session-based RSs for the following two reasons.

Limitation 1. Sequential Dependency Sequence modeling is prevalent for session-based RSs in order to learn sequential dependencies of interactions between users and items. Additive feature attribution does not reflect attribution coming from the sequential order of

[1]Feature attributions typically use the term prediction. In RSs, these predictions are recommendations, and we use these two terms interchangeably in this paper.

Table 6.1: Statistics of the preprocessed dataset (*minimum/median/mean/maximum).

	Diginetica	30Music
Items	3,652	3,261
Sessions	26,176	28,699
Sessions with Repeated Interactions	17,870 (68%)	11,112 (39%)
Session Length*	5/6/7.4/20	5/8/11.6/73
Interactions	194,334	331,653
Interactions Repeated in Sessions	43,997 (23%)	82,655 (25%)

features (i.e. the user-item interactions in session-based RSs), as the attribution scores are not tailored to the position of the features. This means that sequential information, which is key to the recommendation, is lost when computing attribution.

Limitation 2. Repeated Interactions Repeated user interactions occur in session-based RSs when a user interacts with the same item multiple times within a session (note that repeated interactions do not mean that the interactions need to be adjacent in the session). Empirically, this happens frequently in session-based RSs (23%-25% of the times in our two datasets, see Table 6.1). However, additive feature attribution fails to give meaningful scores because these methods rely on independent features, and in the case of repeated interactions, the features are correlated and thereby not independent. So, the assumption of additive feature attribution that features are independent does not hold for session-based RSs.

In the remainder of the paper, we present first an empirical investigation of the extent of the two limitations identified above (in Section 6.4), and then our proposed solution (in Section 6.5).

6.4 Empirical Investigation of the Limitations

We empirically investigate how sequential dependencies and repeated interactions affect explanations of session-based RSs. In this section, we present the experimental setup and our findings. We use publicly available datasets and make our code publicly available.

6.4.1 Experimental Setup

6.4.1.1 Datasets

We experiment with two commonly used datasets for session-based RSs: the Diginetica dataset (we use the item views data[2]) and the 30Music dataset [142]. We use a dataset from the e-commerce domain because users are most often anonymous on e-commerce websites, so session-based RSs are highly relevant for this domain where it is not possible to construct the user profiles typically used in traditional RSs. In addition, we use a dataset from the music domain because music is commonly consumed within listening sessions in sequential order.

[2]`https://competitions.codalab.org/competitions/11161`

Table 6.2: Hyperparamters.

	GRU4REC		SASRec	
	Diginetica	30Music	Diginetica	30Music
Batch Size	128	128	256	256
Units	256	256	64	128
Dropout	0.3	0.4	0.4	0.3

We preprocess the datasets as follows. Note that the datasets do not contain user IDs since the users are anonymous and only identified through their session IDs. We filter out sessions with less than five interactions. We remove items with a frequency less than 20 in the Diginetica dataset and 50 in the larger 30Music dataset such that an item appears in at least 0.01 percent of the sessions. We do this because sparse sessions/items make the evaluation of explanations more unstable, and we are not particularly interested in measuring cold-start recommendation quality in this paper. Finally, we truncate the sessions to have a maximum of 20 interactions in the Diginetica dataset and 73 interactions in the 30Music dataset (the 99th percentiles) to reduce computational cost. The statistics of the datasets after the preprocessing are summarized in Table 6.1. We observe that the Diginetica dataset has slightly more items than the 30Music dataset, while the 30Music dataset has more sessions and interactions. In both datasets, about a quarter of the interactions are repeated interactions, meaning that an interaction with the same item occurs multiple times in the same session. In the Diginetica dataset, the repeated interactions are spread over more sessions (68% of the sessions contain repeated interactions), while in the 30Music dataset, they are concentrated on fewer sessions (39% of the sessions). We split the datasets into training and test sets, such that the test set constitutes the most recent 10 percent of the sessions.

6.4.1.2 Recommendation Models

We experiment with state-of-the-art models for session-based recommendations which we treat as black-box models to be explained. We use GRU4REC [59] to experiment with an RNN-based architecture and SASRec [78] to experiment with a transformer-based architecture. While GRU4REC simply contains a GRU layer, SASRec is a much larger network containing an embedding layer, several self-attention blocks and a prediction layer. We do not experiment with the simple item-to-item recommendation model [36, 90] and Session-based K-Nearest Neighbors [67, 62], as we are interested in black-box models that need explanations. There are extensions of GRU4REC and SASRec that are relevant to explain, for example Neural Attentive Session-based Recommendation [88] that extends GRU4REC with an attention mechanism, and BERT4Rec [132] that extends SASRec with another masking scheme, but the base architectures are the same. We choose GRU4REC and SASRec to experiment with different base architectures that might affect the feature attributions in different ways.

These models have many hyperparameters and it is neither efficient nor customary to tune all of them. We follow the literature [59, 78] and tune only the ones shown in Table 6.2. For the rest, we use default values as per [59, 78]. The optimal hyperparameters found by grid search are presented in Table 6.2. We use early stopping during training with a validation split

Table 6.3: Effectiveness (HR@10 and MRR@10) of the two recommenders per dataset. Best scores are in bold.

Diginetica				
	HR@10	MRR@10	HR@20	MRR@20
GRU4REC	0.4175	0.1592	0.5760	0.1702
SASRec	0.4580	0.1860	**0.6176**	0.1971
30Music				
	HR@10	MRR@10	HR@20	MRR@20
GRU4REC	0.4663	0.3143	0.5204	0.3180
SASRec	**0.4828**	**0.3364**	0.5446	**0.3407**

of the most recent 10 percent of sessions in the training set. The models are typically trained with the ranking losses proposed in GRU4REC and the negative sampling used in SASRec, as these losses are more efficient. However, these losses are not directly compatible with our explainability setup. This is because the models then directly compute rankings of items rather than prediction scores for each item (that can then be used to rank the items). Our explainability setup relies on measuring differences in prediction scores for an individual item. For that reason, we compute a logit score for each item and use the categorical cross-entropy loss. We further added dropout regularization in GRU4REC, as this is suggested in [59] to make it more stable when using cross-entropy loss. SASRec already has dropout regularization. We measure the effectiveness of the models on the last interaction for each session in the test set. We use hit rate (HR) and mean reciprocal rank (MRR), and report the results with a cutoff threshold of 10 and 20 in Table 6.3. We observe that the models generally perform better on the 30Music dataset, which is likely because this dataset is less sparse and generally has longer sessions. We further observe that the SASRec model with most parameters performs best on both datasets.

6.4.1.3 Explainability Methods

We select explainability methods representing three different groups of attribution-based methods: simplification-based, perturbation-based and gradient-based. We present these next. Note that previous work only uses simplification-based and perturbation-based methods to explain RSs (see Section 6.2). We further include gradient-based methods which are designed to explain neural networks, since neural network approaches are common for session-based RSs.

We select the simplification-based technique **LIME** [117]. For each instance in the dataset, LIME trains a linear model to approximate the local decision boundary for that instance. This is done by generating a local dataset consisting of perturbed samples of the instance to explain and the corresponding predictions of the black-box model. On this new dataset, LIME then trains an interpretable model, which is weighted by the proximity of the sampled instances to the instance of interest. Finally, the prediction of interest is explained by interpreting the local model.

We further select the perturbation-based technique **Deep Shap**. It is based on the Shapley Values that compute the average marginal contribution of each feature across all possible feature perturbations. The average marginal contribution of a feature is the average change in the

prediction that the group of features already present receives when the feature value joins them. The Deep variant is a faster approximation of Shapley Values for deep learning models that uses the per-node attribution rules in DeepLIFT to approximate Shapley Values [98].

Finally, we select the gradient-based technique **IG**. IG is a variation of computing the gradient of the model's prediction to its input features. Gradients are the derivative of the output with respect to the inputs. This means that the gradients explain the attribution of features by telling us how small changes of the input features change the output. If the prediction function flattens at the input, it has zero gradient despite the feature value changing the output. IG addresses this issue by computing the path integral of the gradient along a path from a baseline to the input.

We use the explainability methods to explain the top-1 recommendation given to a user. In LIME, we use 5000 samples as the size of the neighborhood to learn the linear model. In Deep Shap, we integrate over the 100 most recent sessions from the training set, as it is only necessary (and most efficient) to integrate over a smaller sample of the training set. In IG, we use as baseline a session with all the interactions replaced by the padding value ignored by the model (e.g. zero), as this should act as an input with no information. We use Deep Shap and IG directly on the logit values (predicted from the model). In LIME, we convert the logit values to probabilities and train local classification models.

In all, we experiment with two different datasets, two different recommendation models, and three different explainability methods resulting in 12 different combinations. The feature attribution methods are computationally expensive. For instance, LIME generates samples and trains a linear model for each instance in the dataset. In addition, it is computationally expensive to investigate sequential dependencies as we do in Section 6.4.2.1 since we need to compare each pairwise combination of interactions in a session. For an average session with 10 interactions that is 45 combinations. For that reason, it is extremely computationally expensive for each dataset, recommendation model and explainability method we include in the experiment, so we focus on datasets and models that best represent the diversity in session-based RSs.

6.4.1.4 Evaluation Procedure

Since we do not have datasets annotated with the ground true explanations, we use the metrics outlined by [1, 13] to indicate the explanations' faithfulness to a model's rationale when no ground truth is available, namely **rank correlation** and **attribute correlation**. We explain these next.

We use **rank correlation** to evaluate how good an explainability method is to rank the importance of the features, where a higher absolute value of attribution scores means higher importance. For each session, we compute rank correlation by first ranking all features by their absolute attribution score, provided by the explainability method. Then we rank the absolute differences in prediction outputs that result from ablating each feature. Finally, we compute Spearman's rank correlation, $\rho(\cdot, \cdot)$, to assess the monotonic relationship between these two rankings. Formally, for a session, s, with interactions $\{i_1, i_2, ..., i_n\}$ and corresponding

attribution scores $\{\phi_1, \phi_2, ..., \phi_n\}$, the rank correlation is given by

$$\text{Rank Correlation} = \rho(\tilde{\boldsymbol{\phi}}, \tilde{\mathbf{d}}), \tag{6.1}$$

where $\tilde{\boldsymbol{\phi}}$ is the vector of feature rankings, $\tilde{\phi}_k$, obtained by taking the absolute value of the attribution scores, that is

$$\tilde{\phi}_k = |\phi_k|, \text{ for } k = 1, ..., n \tag{6.2}$$

and $\tilde{\mathbf{d}}$ is the vector containing the absolute prediction differences (of the top-1 recommendation) given by

$$\tilde{d}_k = |d_k|, \tag{6.3}$$

$$d_k = f(s) - f(s_{\setminus i_k}), \text{ for } k = 1, ..., n, \tag{6.4}$$

where the operation $\setminus i_k$ means all except i_k. In GRU4REC and SASRec, we compute $f(s_{\setminus i_k})$ by setting i_k to a padding value (e.g. zero) that is masked by the model such that the interaction is ignored. The rank correlation measure varies between -1 and 1, with higher being better.

We use **attribute correlation** to evaluate the direction and strength of the attribution scores (e.g. if a feature contributed negatively or positively to the prediction and how much it contributed). We compute the Pearson's correlation, $r(\cdot, \cdot)$, to assess the linear relationship between the attribution of features and the difference in output when ablating those features:

$$\text{Attribute Correlation} = r(\boldsymbol{\phi}, \mathbf{d}) \tag{6.5}$$

where $\boldsymbol{\phi}$ is the vector of attribution scores and $\mathbf{d}$ is defined in Equation 6.4. The attribute correlation measure also varies between -1 and 1, with higher being better.

We compute both rank correlation and attribute correlation for each session in the test set and report the average of the test set.

6.4.2 Experimental Findings

6.4.2.1 Experiment 1. Effect of Sequential Dependencies on Additive Attribution.

We investigate the influence of *sequential dependencies* on additive feature attribution. The problem is that the attribution scores computed by the additive feature attribution methods are not tailored to the position of the features. Thus they should be invariant to the order of interaction, but we posit that this is not the case. We show this by evaluating the explanations when swapping the order of interaction pairs within a session with different degrees of sequential dependencies. Specifically, we evaluate when there is no swap, when we swap the weakest sequential dependency, and when we swap the strongest sequential dependency. For each session in the test set, we detect sequential dependencies by measuring the difference in prediction output when swapping each combination of interaction pairs. That is, for a pair of interactions $\{i_k, i_l\}$, we quantify the difference as

$$|f(.., i_k, .., i_l, ..) - f(.., i_l, .., i_k, ..)|, \tag{6.6}$$

Table 6.4: Influence of sequential dependencies on additive feature attribution. Percentages in brackets denote the difference from no swap. No swap uses the original sequence of interactions in the session. Swap weakest/strongest dependency swaps the order of the weakest/strongest interactions in the session.

(a) Diginetica

	GRU4REC		SASRec	
	LIME		LIME	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.1806	0.1373	0.3103	0.8142
Swap weakest dependency	0.1475 (-18.35%)	0.1236 (-10.00%)	0.2032 (-34.52%)	0.5569 (-31.60%)
Swap strongest dependency	0.0127 (-92.98%)	0.0817 (-40.52%)	-0.0169 (-105.44%)	-0.0854 (-110.49%)
	Deep Shap		Deep Shap	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.4207	0.2857	0.4571	0.9326
Swap weakest dependency	0.3424 (-18.59%)	0.2510 (-12.13%)	0.3203 (-29.92%)	0.6464 (-30.69%)
Swap strongest dependency	0.0691 (-83.58%)	0.1924 (-32.64%)	-0.0208 (-104.54%)	-0.1304 (-113.99%)
	IG		IG	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.4060	0.3683	0.5439	0.9528
Swap weakest dependency	0.3358 (-17.27%)	0.3272 (-11.14%)	0.3785 (-30.41%)	0.6620 (-30.52%)
Swap strongest dependency	0.0783 (-80.71%)	0.2488 (-32.44%)	-0.0162 (-102.97%)	-0.1261 (-113.24%)

(b) 30Music

	GRU4REC		SASRec	
	LIME		LIME	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.1420	0.2404	0.2563	0.7906
Swap weakest dependency	0.1335 (-5.98%)	0.2239 (-6.85%)	0.2213 (-13.67%)	0.6644 (-15.96%)
Swap strongest dependency	0.0172 (-87.86%)	0.1326 (-44.84%)	0.0160 (-93.75%)	0.0473 (-94.02%)
	Deep Shap		Deep Shap	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.4911	0.6327	0.3784	0.9056
Swap weakest dependency	0.4405 (-10.30%)	0.5901 (-6.73%)	0.3354 (-11.38%)	0.7524 (-16.92%)
Swap strongest dependency	0.1298 (-73.57%)	0.3621 (-42.77%)	-0.0405 (-110.70%)	0.0531 (-94.14%)
	IG		IG	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
No swap	0.5181	0.7341	0.4240	0.9352
Swap weakest dependency	0.4743 (-8.46%)	0.6905 (-5.95%)	0.3800 (-10.36%)	0.7779 (-16.82%)
Swap strongest dependency	0.1551 (-70.06%)	0.4298 (-41.45%)	-0.0097 (-102.29%)	0.0549 (-94.13%)

where $|\cdot|$ denotes the absolute value. The pair of interactions that obtains the greatest value in Equation 6.6 for a session is then defined as the strongest sequential dependency for that session, and the pair of interactions that obtains the value closest to zero in Equation 6.6 is defined as the weakest sequential dependency for that session.

The results are presented in Table 6.4. Across all the explainability techniques, both datasets and both models, we observe less faithful attribution scores, in terms of lower rank correlation and attribute correlation, when we swap the order of the stronger sequential dependencies in a session (up to -113% in correlation drop). Further, we observe a drop in the correlation (hence less faithful attribution) even when we swap the order of the weakest dependencies, compared to the original sequence. Especially Deep Shap and IG with SASRec seem to have the most faithful attribution scores when there is no swap, but they have the largest drop in rank correlation and

Table 6.5: Influence of repeated interactions on additive feature attribution. Percentages in brackets denote the difference from all types of interactions.

(a) Diginetica

	GRU4REC		SASRec	
	LIME		LIME	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.1863	0.1034	0.3062	0.4485
Repeated interactions	0.1330 (-28.62%)	0.0551 (-46.73%)	0.2551 (-16.70%)	0.4212 (-6.08%)
	Deep Shap		Deep Shap	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.4421	0.2273	0.4789	0.5584
Repeated interactions	0.3617 (-18.17%)	0.1266 (-44.32%)	0.4772 (-0.36%)	0.5196 (-6.96%)
	IG		IG	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.4163	0.3518	0.5167	0.7320
Repeated interactions	0.3325 (-20.13%)	0.1844 (-47.59%)	0.5019 (-2.86%)	0.6291 (-14.06%)

(b) 30Music

	GRU4REC		SASRec	
	LIME		LIME	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.0968	0.1390	0.1885	0.4483
Repeated interactions	0.0967 (-0.14%)	0.1032 (-25.80%)	0.1510 (-19.90%)	0.4459 (-0.54%)
	Deep Shap		Deep Shap	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.4754	0.4218	0.3660	0.6236
Repeated interactions	0.4001 (-15.84%)	0.2892 (-31.45%)	0.3652 (-0.20%)	0.5556 (-10.89%)
	IG		IG	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
All types of interactions	0.4827	0.5478	0.4046	0.6626
Repeated interactions	0.4042 (-16.25%)	0.3465 (-36.74%)	0.4078 (0.81%)	0.5381 (-18.79%)

attribute correlation when swapping the sequential dependencies. This is likely because SASRec better captures sequential dependencies than GRU4REC.

Overall, the impact of sequential dependency to feature attribution faithfulness is notable, meaning that the order of interactions within sessions cannot be ignored in session-based RS explainability.

6.4.2.2 Experiment 2. Effect of Repeated Interactions on Additive Feature Attribution

We further investigate how additive feature attribution is affected by the *repeated user interactions* that occur in session-based RSs. The problem is that all features are handled as independent features by the additive feature attribution methods, but repeated interactions are correlated and thereby not independent. We investigate this by evaluating the explanations solely over repeated interactions compared to evaluating the explanations over all types of interactions that can be both repeated and non-repeated. Since the evaluation measures are affected by sample size, we want to evaluate over the same number of interactions in the two groups that we compare. For that reason, we randomly sample from the group with all types of interactions,

such that this group has the same sample size as the group with repeated interactions.

The results are presented in Table 6.5. Overall, the faithfulness of feature attribution is notably lower in the data subset that contains only repeated interactions than in the original data, across the different explainability techniques, models and datasets. This is especially the case for the attribute correlation. This is likely because it has a different effect on the prediction score when a user interacts with the same item twice compared to once. The largest drop in faithfulness on repeated interactions compared to all types of interactions is seen with GRU4REC. It is presumably because the repeated interactions are less correlated in SASRec, since they have different positional embeddings.

Overall, the impact of repeated interactions to feature attribution faithfulness is notable, meaning that repeated interactions cannot be ignored in session-based RS explainability.

6.5 Proposed Explainability Approach

In this section, we propose a simple method to address the above problems with sequential dependencies and repeated interactions, and we evaluate our method against the additive feature attribution methods.

6.5.1 Our Session-based Occlusion Method

In order to address the problem of explaining session-based recommendations, we propose an approach that computes joint attribution scores for sets of interactions with sequential dependencies and sets of repeated interactions, since these feature sets represent non-linear relations. With standard additive feature attribution, the attribution score of a set of features is computed as the sum of their individual scores. Differently from this, we choose an occlusion-based explainability method that we can adapt so that we can compute non-additive attributions of feature sets. Occlusion-based approaches compute the attribution score of a feature by measuring the prediction difference caused by ablating that particular feature. Instead of ablating each feature individually, we ablate sets of interactions with sequential dependencies or with repeated interactions at the same time, thereby measuring the joint attribution from these specific sets of features.

We extend the occlusion-based method called Randomized Input Sampling for Explanation (RISE) [109], as it is efficient and context-independent. RISE was originally designed to explain image recognition models. The idea is to probe the base model by sub-sampling the input image via random masks and recording the prediction differences in each of the masked images. The final attribution map is generated as the average of the prediction differences weighted by the masks. We adapt RISE to fit our scenario as follows. For a session $s = \{i_1, i_2, ..., i_n\}$, let $\{m_1, ...m_M\}$ be M random sub-samples of the interactions in s. The attribution score of an interaction i_k is given by

$$\phi_k = \frac{1}{|S_k|} \sum_{m_j \in S_k} f(s) - f(s_{\setminus m_j}), \tag{6.7}$$

Table 6.6: Our approach compared to additive feature attribution. Best scores are in bold. The percentage difference of our method from the strongest and weakest baseline respectively is presented in parentheses.

(a) Diginetica

	GRU4REC		SASRec	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
LIME	0.2502	0.2150	0.4839	0.8151
Deep Shap	0.3691	0.2924	0.5774	0.8838
IG	0.3814	0.3439	0.6551	0.9358
Session-based Occlusion	**0.4025** (6%-61%)	**0.6425** (87%-199%)	**0.7076** (8%-46%)	**0.9504** (2%-17%)

(b) 30Music

	GRU4REC		SASRec	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
LIME	0.2095	0.2711	0.4411	0.6655
Deep Shap	0.3908	0.4474	0.4748	0.6729
IG	0.4160	0.5232	0.4878	0.6862
Session-based Occlusion	**0.4261** (2%-103%)	**0.6328** (21%-133%)	**0.5276** (8%-20%)	**0.7286** (6%-9%)

where $S_k = \{m_j | i_k \in m_j\}$ is the set of sub-samples containing i_k and $|S_k|$ denotes the cardinality of S_k. We extend the above to jointly compute an attribution score of a set of interactions:

$$\phi_{\mathcal{I}} = \frac{1}{|S_{\mathcal{I}}|} \sum_{m_j \in S_{\mathcal{I}}} f(s) - f(s_{\setminus m_j}), \tag{6.8}$$

where $S_{\mathcal{I}} = \{m_j | \mathcal{I} \in m_j\}$ and $\mathcal{I} \subseteq \{1, 2, .., n\}$ is a set of interaction indices.

We call our method *session-based occlusion* and it comprises two steps. (1) Detect sets of interactions with sequential dependencies and sets of repeated interactions; (2) Use the occlusion-based method to compute joint attribution scores for the detected sets of interactions and individual attribution scores for the rest. In (1), we detect sets of interactions with sequential dependencies by using Equation 6.6. Theoretically, there should be a sequential dependency if the difference is greater than zero. However, we require the difference to be larger than a small threshold of 2 to avoid negligible sequential dependencies. As opposed to the experiment in Section 6.4.2.1, where we simply wanted to swap interactions that cause the smallest and largest prediction difference, respectively, we now want to detect exactly which pairs of interactions cause prediction differences when swapping the order. If we, for example, have a session with interactions $\{i_1, i_2, i_3, i_4, i_5\}$ and we swap the position of i_1 and i_3, the order of i_1 and i_2 as well as i_2 and i_3 will also change. For that reason, we replace all other interactions than the two we are investigating with the padding value that is masked by the model (e.g. zero) to measure the isolated prediction difference when detecting sequential dependencies. Moreover, we merge overlapping pairs, for example, if interaction i_1 and i_2 have a sequential dependency and interaction i_2 and i_3 (or i_1 and i_3) have a sequential dependency, we compute a joint attribution score for the set $\{i_1, i_2, i_3\}$. Finally, we select only repeated interactions that are not in the sets with sequential dependencies. In (2), we use Equation 6.8 to compute the attribution scores.

Table 6.7: Effect of joint attribution scores: We evaluate our approach when attribution scores are computed individually for each feature.

(a) Diginetica

	GRU4REC		SASRec	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
Session-based Occlusion without Joint Attributions	0.3491	0.4600	0.5760	0.7822
Session-based Occlusion	0.4025	0.6425	0.7076	0.9504

(b) 30Music

	GRU4REC		SASRec	
	Rank Correlation	Attribute Correlation	Rank Correlation	Attribute Correlation
Session-based Occlusion without Joint Attributions	0.3707	0.5460	0.4960	0.6942
Session-based Occlusion	0.4261	0.6328	0.5276	0.7286

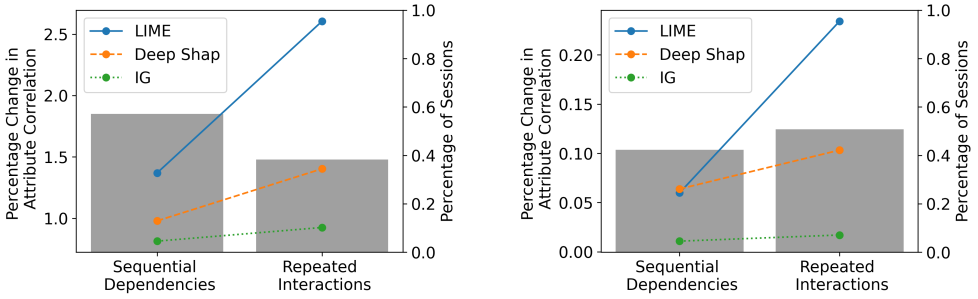
6.5.2 Results

We compare our approach against the additive feature attribution methods evaluated in Section 6.4.2 (Table 6.4 and 6.5), where the attribution scores of the detected sets of sequential dependencies and repeated interactions are computed as the sum of their individual scores. The results are presented in Table 6.6. We see that our session-based occlusion method computes the most faithful attribution scores across both models and datasets. In Section 6.4.2, we saw that explanations of GRU4REC were the most affected by the repeated interactions, while explanations of SASRec were the most affected by the sequential dependencies, and we see an improvement on both now. The biggest improvement is seen in the attribute correlation on GRU4REC, which was also found to be most affected by the repeated interactions in Section 6.4.2.

To further test the effect of computing the joint attribution scores of the sequential dependencies and repeated interactions, we evaluate our method when the scores are computed individually (i.e. with Equation 6.7 instead of Equation 6.8). We do this to measure the isolated effect of computing joint attribution. The results are presented in Table 6.7. We observe that our method is considerably more faithful when jointly computing the attribution scores of the sequential dependencies and repeated interactions.

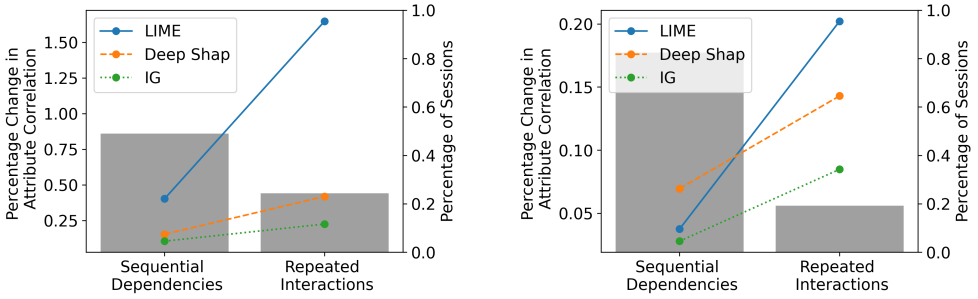
6.5.3 Analysis of Results

We further break down the faithfulness of our method to see how it compares to LIME, Deep Shap and IG on sessions with sequential dependencies and sessions with repeated interactions. We do that by measuring the percentage change in attribute correlation of our method over LIME, Deep Shap and IG. A high percentage change means that the attribute correlation of our method is better than the one we compare against. We break down the analysis on sessions in the test set containing sequential dependencies and repeated interactions detected by our method. The analysis is presented in Figure 6.2. We observe that our method generally has the greatest improvement over LIME, Deep Shap and IG on the sessions containing repeated interactions. On the other hand, there are generally more sessions with sequential dependencies than sessions with repeated interactions (with the exception of the SASRec model on the Diginetica dataset),



GRU4REC SASRec

(a) Diginetica



GRU4REC SASRec

(b) 30Music

Figure 6.2: Percentage change in attribute correlation of our method over LIME, Deep Shap and IG (left axis). The grey bars indicate the percentage of sessions in the test set containing sequential dependencies and repeated interactions detected by our method (right axis). Note that our method selects only repeated interactions that are not in the sets with sequential dependencies. When a session contains both sequential dependencies and repeated interactions, it is included in both groups.

so the improvement on sequential dependencies affects more sessions. LIME generally has the biggest gap between the two groups, while IG has the smallest gap. For example with the SASRec model on the Diginetica dataset, the attribute correlation of our method is approximately 6% greater than LIME on the sessions with sequential dependencies but more than 20% greater on the sessions with repeated interactions, while the attribute correlation of our method is approximately 2% greater than IG on both the sessions with sequential dependencies and the sessions with repeated interactions.

Next, we find examples where the faithfulness of our method is better than LIME, Deep Shap and IG to illustrate how the difference affects the resulting explanations. Figure 6.3 presents an example with attribution scores (normalized) generated for a session with a sequential dependency between interaction i_1 and i_2 (detected by our method). The session is an example

	Sequential Dependency				Rank Correlation	Attribute Correlation
	i_1	i_2	i_3	i_4		
LIME	-1.60		1.17	0.43	0.33	-0.96
	(-0.50, -1.10)					
Deep Shap	-1.28		-0.18	1.46	-1.00	-0.40
	(-0.71, -0.57)					
IG	-1.16		-0.33	1.49	-1.00	-0.30
	(-0.60, -0.56)					
Session-based Occlusion	1.00		-1.01	0.01	1.00	1.00

Figure 6.3: Example of attribution scores (normalized) generated for a session with a sequential dependency between two interactions. For LIME, Deep Shap and IG, the attribution score for the sequential dependency is computed as the sum of their individual scores indicated in parentheses.

			Repeated Interaction				Rank Correlation	Attribute Correlation
	i_1	i_2	i_3	i_4	i_5	i_6		
LIME	-0.31	-0.44	-0.97		-0.32	2.04	0.40	0.97
			(-0.54, -0.43)					
Deep Shap	-0.52	-0.49	-0.78		-0.19	1.98	0.00	0.98
			(-0.41, -0.37)					
IG	-0.34	-0.43	-0.90		-0.37	2.04	0.40	0.98
			(-0.45, -0.45)					
Session-based Occlusion	-0.43	-0.45	-0.44		-0.46	1.78	1.00	1.00

Figure 6.4: Example of attribution scores (normalized) generated for a session with a pair of repeated interactions (i.e. interaction i_3 and i_4 is with the same item). For LIME, Deep Shap and IG, the attribution score for the repeated interactions is computed as the sum of their individual scores indicated in parentheses.

from the Diginetica dataset and the GRU4REC model. For LIME, Deep Shap and IG, the attribution score for the interactions with a sequential dependency is computed as the sum of their individual scores. We see that all three methods find that the individual scores for these two interactions are negative, but the joint attribution score computed by our method is positive. This is likely because the isolated contribution to the prediction of these two interactions is negative, but the model has learned a sequential dependency, so when they appear in this particular order, they have a positive contribution to the prediction. We see that this also affects the attribution score of the other interactions in the session, as the attribution scores for i_3 and i_4 found by LIME, Deep Shap and IG are very different from the ones found by our method. This is because the additive feature attribution methods try to distribute the contributions between the interactions. In this example, both the rank correlation and the attribute correlation are worse with LIME, Deep Shap and IG than with our method. Figure 6.4 presents an example of attribution scores (normalized) generated for a session where i_3 and i_4 are interactions with the same item (i.e. repeated interactions). This session is an example from the Diginetica dataset and the SASRec model. We see that LIME, Deep Shap and IG assign to each of the repeated

interactions approximately the same contribution as our method assigns to both of them in total. This is likely because this interaction has the same contribution to the prediction no matter how many times it appears in the session. This is captured by our method, but not by the standard additive feature attribution methods. In this example, the attribute correlation is almost the same for all the methods, but the rank correlation is lower with LIME, Deep Shap and IG than with our method, because they assign too much importance to the repeated interactions.

6.6 Conclusion

In this work, we study feature attribution methods for explaining session-based recommendations. These are highly popular explainability methods that work post hoc without affecting recommendation accuracy, but they have not yet been adapted to the session-based recommendation task. We identify two limitations of using existing feature attribution methods to explain session-based recommendations, namely that these methods do not account for (1) sequential dependencies and (2) repeated user interactions that are present in session-based RSs. The consequences are unfaithful explanations that do not reflect the attribution coming from sequential dependencies and correlated user interactions. We further present a simple feature attribution method that overcomes these limitations. Our method uses occlusion to incorporate non-additive attribution scores for sets of interactions with sequential dependencies and sets of repeated interactions. Experimental results on real-life datasets and session-based recommendation models show that our method outperforms state-of-the-art feature attribution methods notably. Our work paves the way for further research on feature attribution explanations for session-based recommendations.

Bibliography

[1] Chirag Agarwal, Satyapriya Krishna, Eshika Saxena, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. OpenXAI: Towards a Transparent Evaluation of Model Explanations. In *Advances in Neural Information Processing Systems*, volume 35, pages 15784–15799. Curran Associates, Inc., 2022. URL https://openreview.net/forum?id=MU2495w47rz.

[2] Charu C. Aggarwal. *Context-Sensitive Recommender Systems*, pages 255–281. Springer International Publishing, 2016. URL https://doi.org/10.1007/978-3-319-29659-3_8.

[3] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[4] Charu C. Aggarwal. *Content-Based Recommender Systems*, pages 139–166. Springer International Publishing, 2016.

[5] Karan Aggarwal, Pranjul Yadav, and S. Sathiya Keerthi. Domain Adaptation in Display Advertising: an Application for Partner Cold-start. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 178–186. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3298689.3347004.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995. URL https://api.semanticscholar.org/CorpusID:12242182.

[7] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Antonio Ferrara, and Alberto Carlo Maria Mancino. Sparse Feature Factorization for Recommender Systems with Knowledge Graphs. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pages 154–165. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3460231.3474243.

[8] Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. Diagnostics-Guided Explanation Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10445–10453, 2022. URL https://ojs.aaai.org/index.php/AAAI/article/view/21287.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

[10] Jie Bao and Yu Zheng. *Location-Based Recommendation Systems*, pages 1145–1153. Springer International Publishing, 2017. URL https://doi.org/10.1007/978-3-319-17885-1_1580.

[11] Oren Barkan, Noam Koenigstein, Eylon Yogev, and Ori Katz. CB2CF: a Neural Multiview Content-to-Collaborative Filtering Model for Completely Cold Item Recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 228–236. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3298689.3347038.

[12] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval*, ISMIR '11, 2011.

[13] Umang Bhatt, Adrian Weller, and José M. F. Moura. Evaluating and Aggregating Feature-based Model Explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3016–3022. International Joint Conferences on Artificial Intelligence Organization, 2020. URL https://doi.org/10.24963/ijcai.2020/417.

[14] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. DCDIR: A Deep Cross-Domain Recommendation System for Cold Start Users in Insurance Domain. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 1661–1664. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3397271.3401193.

[15] Mirko Biasini. Design and Implementation of Gamification in a Social e-Learning Platform for Increasing Learner Engagement. Master's thesis, Danmarks Tekniske Universitet and Università degli Studi di Padova, 2020.

[16] Toine Bogers, Cataldo Musto, David Wang, Alexander Felfernig, Simone Borg Bruun, Giovanni Semeraro, and Yong Zheng. FinRec: The 3rd International Workshop on Personalization & Recommender Systems in Financial Services. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 688–690. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3523227.3547420.

[17] Simone Borg Bruun. Learning Dynamic Insurance Recommendations from Users' Click Sessions. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 860–863. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3460231.3473900.

[18] Antoine Brenner, Bruno Pradel, Nicolas Usunier, and Patrick Gallinari. Predicting Most Rated Items in Weekly Recommendation with Temporal Regression. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, page 24–27. Association for Computing Machinery, 2010. URL https://doi.org/10.1145/1869652.1869656.

[19] Léo Brunot, Nicolas Canovas, Alexandre Chanson, Nicolas Labroche, and Willème Verdeaux. Preference-based and Local Post-hoc Explanations for Recommender Systems. *Information Systems*, 108, 2022. URL https://www.sciencedirect.com/science/article/pii/S0306437922000254.

[20] Simone Borg Bruun, Maria Maistro, and Christina Lioma. Learning Recommendations from User Actions in the Item-poor Insurance Domain. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pages 113–123. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3523227.3546775.

[21] Simone Borg Bruun*, Kacper Kenji Leśniak*, Mirko Biasini, Vittorio Carmignani, Panagiotis Filianos, Christina Lioma, and Maria Maistro. Graph-Based Recommendation for Sparse and Heterogeneous User Interactions. In *Advances in Information Retrieval*, ECIR '23, pages 182–199. Springer Nature Switzerland, 2023. URL https://doi.org/10.1007/978-3-031-28244-7_12.

[22] Simone Borg Bruun, Christina Lioma, and Maria Maistro. Recommending Target Actions Outside Sessions in the Data-Poor Insurance Domain. *ACM Transactions on Recommer Systems*, 2023. URL https://doi.org/10.1145/3606950.

[23] Simone Borg Bruun, Krisztian Balog, and Maria Maistro. Dataset and Models for Item Recommendation Using Multi-Modal User Interactions. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24. Association for Computing Machinery, 2024.

[24] Simone Borg Bruun, Christina Lioma, and Maria Maistro. Feature Attribution Explanations of Session-based Recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24. Association for Computing Machinery, 2024.

[25] Desheng Cai, Shengsheng Qian, Quan Fang, Jun Hu, and Changsheng Xu. Adaptive Anti-Bottleneck Multi-Modal Graph Learning Network for Personalized Micro-video Recommendation. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 581–590. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3503161.3548420.

[26] Lei Cai, Zhengyang Wang, Hongyang Gao, Dinggang Shen, and Shuiwang Ji. Deep Adversarial Learning for Multi-Modality Missing Data Completion. In *Proceedings of the*

24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, page 1158–1166. Association for Computing Machinery, 2018. URL https://doi.org/10.1145/3219819.3219963.

[27] Chun-Hao Kingsley Chang, Elliot Creager, Anna Goldenberg, and David Kristjanson Duvenaud. Explaining Image Classifiers by Counterfactual Generation. In *International Conference on Learning Representations*, 2018. URL https://api.semanticscholar.org/CorpusID:52962991.

[28] Feiyu Chen, Junjie Wang, Yinwei Wei, Hai-Tao Zheng, and Jie Shao. Breaking Isolation: Multimodal Graph Fusion for Multimedia Recommendation by Edge-wise Modulation. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 385–394, New York, NY, USA, 2022. Association for Computing Machinery. URL https://doi.org/10.1145/3503161.3548399.

[29] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Transactions on Information Systems*, 41(3), 2023. URL https://doi.org/10.1145/3564284.

[30] Jiayi Chen, Wen Wu, Wenxin Hu, Wei Zheng, and Liang He. SSR: Explainable Session-based Recommendation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9534196.

[31] Li Chen and Weike Pan. CoFiSet: Collaborative Filtering via Learning Pairwise Preferences over Item-sets. In *SIAM International Conference on Data Mining*, 2013. URL https://api.semanticscholar.org/CorpusID:14753641.

[32] Tianle Chen, Brian Keng, and Javier Moreno. Multivariate Arrival Times with Recurrent Neural Networks for Personalized Demand Forecasting. *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 810–819, 2018.

[33] Xiancong Chen, Lin Li, Weike Pan, and Zhong Ming. A Survey on Heterogeneous One-class Collaborative Filtering. *ACM Transactions on Information Systems*, 38(4), 2020. URL https://doi.org/10.1145/3402521.

[34] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST@EMNLP '14, pages 103–111. Association for Computational Linguistics, 2014. URL https://aclanthology.org/W14-4012/.

[35] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, pages 39–46. Association for Computing Machinery, 2010. URL https://doi.org/10.1145/1864708.1864721.

[36] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The YouTube Video Recommendation System. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, pages 293–296. Association for Computing Machinery, 2010. URL https://doi.org/10.1145/1864708.1864770.

[37] Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998. URL https://doi.org/10.1162/089976698300017197.

[38] Jingtao Ding, Guanghui Yu, Xiangnan He, Fuli Feng, Yong Li, and Depeng Jin. Sampler Design for Bayesian Personalized Ranking by Leveraging View Data. *IEEE Transactions on Knowledge and Data Engineering*, 33:667–681, 2018. URL https://api.semanticscholar.org/CorpusID:52343524.

[39] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. Improving Implicit Recommender Systems with View Data. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3343–3349. International Joint Conferences on Artificial Intelligence Organization, 2018. URL https://doi.org/10.24963/ijcai.2018/464.

[40] Jie Dong, Gui Li, Wenkai Ma, and Jianshun Liu. Personalized Recommendation System based on Social Tags in the Era of Internet of Things. *Journal of Intelligent Systems*, 31 (1):681–689, 2022. URL https://doi.org/10.1515/jisys-2022-0053.

[41] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant Colony Optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006. URL https://doi.org/10.1007/978-0-387-30164-8_22.

[42] Nan Du, Yichen Wang, Niao He, and Le Song. Time-Sensitive Recommendation from Recurrent User Activities. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS '15, page 3492–3500. MIT Press, 2015.

[43] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Transactions on Information Systems*, 39(1):10:1–10:42, 2020. URL https://doi.org/10.1145/3426723.

[44] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM Transactions on Information Systems*, 39(2):20:1–20:49, 2021. URL https://doi.org/10.1145/3434185.

[45] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In

Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '19, pages 1069–1072. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3331184.3331322.

[46] Anna Gentile, Vitaveska Lanfranchi, Suvodeep Mazumdar, and Fabio Ciravegna. Extracting Semantic User Networks from Informal Communication Exchanges. In *The Semantic Web – ISWC 2011*, pages 209–224. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-25073-6_14.

[47] The World Bank Group. Small and Medium Enterprises (SMEs) Finance, 2022. URL https://www.worldbank.org/en/topic/smefinance. Last accessed: 2022-10-04.

[48] Asela Gunawardana and Guy Shani. *Evaluating Recommender Systems*, pages 265–308. Springer US, 2015. URL https://doi.org/10.1007/978-1-4899-7637-6_8.

[49] Aaron Halfaker, Os Keyes, Daniel Kluver, Jacob Thebault-Spieker, Tien Nguyen, Kate Grandprey-Shores, Anuradha Uduwage, and Morten Warncke-Wang. User Session Identification Based on Strong Regularities in Inter-activity Time. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 410–418. Association for Computing Machinery, 2015. URL https://doi.org/10.1145/2736277.2741117.

[50] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. Content-Aware Neural Hashing for Cold-Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 971–980. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3397271.3401060.

[51] Christian Hansen, Casper Hansen, Niklas Hjuler, Stephen Alstrup, and Christina Lioma. Sequence Modelling For Analysing Student Interaction with Educational Systems. In *EDM*. International Educational Data Mining Society (IEDMS), 2017.

[52] Christian Hansen, Casper Hansen, Jakob Grue Simonsen, and Christina Lioma. Projected Hamming Dissimilarity for Bit-Level Importance Coding in Collaborative Filtering. In *Proceedings of the Web Conference 2021*, WWW '21, page 261–269. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3442381.3450011.

[53] Ghazaleh Haratinezhad Torbati, Andrew Yates, and Gerhard Weikum. You Get What You Chat: Using Conversations to Personalize Search-based Recommendations. In *Advances in Information Retrieval: ECIR 2021*, pages 207–223. Springer International Publishing, 2021.

[54] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4):19:1–19:19, 2016. URL https://doi.org/10.1145/2827872.

[55] Ruining He and Julian McAuley. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI '16, page 144–150. AAAI Press, 2016.

[56] Ruining He and Julian McAuley. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *Proceedings of the IEEE 16th International Conference on Data Mining*, ICDM '16, pages 191–200. IEEE Computer Society, 2016. URL `https://doi.org/10.1109/ICDM.2016.0030`.

[57] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182. Association for Computing Machinery, 2017. URL `https://doi.org/10.1145/3038912.3052569`.

[58] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, page 241–250, 2000. URL `https://doi.org/10.1145/358916.358995`.

[59] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations*, ICLR '16, 2016. URL `http://arxiv.org/abs/1511.06939`.

[60] Balázs Hidasi and Alexandros Karatzoglou. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 843–852. Association for Computing Machinery, 2018. URL `https://doi.org/10.1145/3269206.3271761`.

[61] John H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–73, 1992. URL `http://www.jstor.org/stable/24939139`.

[62] Haoji Hu, Xiangnan He, Jinyang Gao, and Zhi-Li Zhang. Modeling Personalized Item Frequency Information for Next-basket Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 1071–1080. Association for Computing Machinery, 2020. URL `https://doi.org/10.1145/3397271.3401066`.

[63] Liang Hu, Qingkui Chen, Haiyan Zhao, Songlei Jian, Longbing Cao, and Jian Cao. Neural Cross-Session Filtering: Next-Item Prediction Under Intra- and Inter-Session Context. *IEEE Intelligent Systems*, pages 57–67, 2018.

[64] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008. URL `https://doi.ieeecomputersociety.org/10.1109/ICDM.2008.22`.

[65] Bei Hui, Lizong Zhang, Xue Zhou, Xiao Wen, and Yuhui Nian. Personalized Recommendation System based on Knowledge Embedding and Historical Behavior. *Applied Intelligence*, 52(1):954–966, 2022. URL https://doi.org/10.1007/s10489-021-02363-w.

[66] Laura Inozemtseva, Reid Holmes, and Robert J. Walker. Recommendation Systems in-the-Small. In *Recommendation Systems in Software Engineering*, pages 77–92. Springer, 2014. URL https://doi.org/10.1007/978-3-642-45135-5_4.

[67] Dietmar Jannach and Malte Ludewig. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 306–310. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3109859.3109872.

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

[69] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Knowledge-based Recommendation*, page 81–123. Cambridge University Press, 2010.

[70] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. A Survey on Conversational Recommender Systems. *ACM Computing Surveys*, 54(5), 2021. URL https://doi.org/10.1145/3453154.

[71] Bernard J. Jansen, Amanda Spink, and Vinish Kathuria. How to Define Searching Sessions on Web Search Engines. In *Advances in Web Mining and Web Usage Analysis, proceedings of 8th International Workshop on Knowledge Discovery on the Web*, volume 4811 of *WebKDD '06*, pages 92–109. Springer, 2006. URL https://doi.org/10.1007/978-3-540-77485-3_6.

[72] Bernard J. Jansen, Amanda Spink, Chris Blakely, and Sherry Koshman. Defining a Session on Web Search Engines. *Journal of the American Society for Information Science and Technology*, 58(6):862–871, 2007. URL https://doi.org/10.1002/asi.20564.

[73] How Jing and Alexander J. Smola. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, page 515–524. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3018661.3018719.

[74] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, page 154–161. Association for Computing Machinery, 2005. URL https://doi.org/10.1145/1076034.1076063.

[75] Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. Representation of Linguistic Form and Function in Recurrent Neural Networks. *Computational Linguistics*, 43(4):761–780, 2017. URL https://aclanthology.org/J17-4003.

[76] Marius Kaminskas, Derek Bridge, Franclin S. Foping, and Donogh Roche. Product Recommendation for Small-Scale Retailers. In *Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies*, volume 239 of *EC-Web '15*, pages 17–29. Springer, 2015. URL https://doi.org/10.1007/978-3-319-27729-5_2.

[77] Marius Kaminskas, Derek Bridge, Franclin S. Foping, and Donogh Roche. Product-Seeded and Basket-Seeded Recommendations for Small-Scale Retailers. *Journal on Data Semantics*, 6(1):3–14, 2017. URL https://doi.org/10.1007/s13740-016-0058-3.

[78] Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE Computer Society, 2018. URL https://doi.ieeecomputersociety.org/10.1109/ICDM.2018.00035.

[79] James Kennedy and Russell Eberhart. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995. URL https://doi.org/10.1007/978-0-387-30164-8_630.

[80] Zühal Kurt and Kemal Özkan. An Image-based Recommender System based on Feature Extraction Techniques. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 769–774, 2017.

[81] M.H. Kutner, , C. J. Nachtsheim, J. Neter, W. Li, et al. *Applied Linear Statistical Models*. McGraw-Hill, Irwin, 2005.

[82] Urszula Kuzelewska. Effect of Dataset Size on Efficiency of Collaborative Filtering Recommender Systems with Multi-clustering as a Neighbourhood Identification Strategy. In *Proceedings of the 20th International Conference on Computational Science*, volume 12139 of *ICCS '20*, pages 342–354. Springer, 2020. URL https://doi.org/10.1007/978-3-030-50420-5_25.

[83] Sara Latifi, Noemi Mauro, and Dietmar Jannach. Session-aware Recommendation: A Surprising Quest for the State-of-the-art. *Information Sciences*, 573:291–315, 2021. URL https://doi.org/10.1016/j.ins.2021.05.048.

[84] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. In *Proceedings of the 44th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 1513–1522. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3404835.3462935.

[85] Sangkeun Lee, Sungchan Park, Minsuk Kahng, and Sang-goo Lee. PathRank: Ranking Nodes on a Heterogeneous Graph for Flexible Hybrid Recommender Systems. *Expert Systems with Applications*, 40(2):684–697, 2013. URL https://doi.org/10.1016/j.eswa.2012.08.004.

[86] Yen-Hsien Lee, Tsang-Hsiang Cheng, Ci-Wei Lan, Chih-Ping Wei, and Paul Jen-Hwa Hu. Overcoming Small-size Training Set Problem in Content-based Recommendation: a Collaboration-based Training Set Expansion Approach. In *Proceedings of the 11th International Conference on Electronic Commerce*, ICEC '09, pages 99–106. Association for Computing Machinery, 2009. URL https://doi.org/10.1145/1593254.1593268.

[87] Lukas Lerche and Dietmar Jannach. Using Graded Implicit Feedback for Bayesian Personalized Ranking. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 353–356. Association for Computing Machinery, 2014. URL https://doi.org/10.1145/2645710.2645759.

[88] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural Attentive Session-based Recommendation. In *Proceedings of the 26th ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1419–1428. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3132847.3132926.

[89] Ran Li, Yuexin Li, Jingsheng Lei, and Shengying Yang. A Multi-Behavior Recommendation Method for Users Based on Graph Neural Networks. *Applied Sciences*, 13(16), 2023. URL https://www.mdpi.com/2076-3417/13/16/9315.

[90] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[91] Fan Liu, Zhiyong Cheng, Changchang Sun, Yinglong Wang, Liqiang Nie, and Mohan Kankanhalli. User Diverse Preference Modeling by Multimodal Attentive Metric Learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 1526–1534. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3343031.3350953.

[92] Shang Liu, Zhenzhong Chen, Hongyi Liu, and Xinghai Hu. User-Video Co-Attention Network for Personalized Micro-video Recommendation. In *The World Wide Web Conference*, WWW '19, page 3020–3026. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3308558.3313513.

[93] Siyuan Liu, Qiong Wu, and Chunyan Miao. Personalized Recommendation Considering Secondary Implicit Feedback. In *2018 IEEE International Conference on Agents (ICA)*, pages 87–92. IEEE Computer Society, 2018. URL https://doi.ieeecomputersociety.org/10.1109/AGENTS.2018.8460053.

[94] Xiaohao Liu, Zhulin Tao, Jiahong Shao, Lifang Yang, and Xianglin Huang. EliMRec: Eliminating Single-modal Bias in Multimedia Recommendation. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 687–695. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3503161.3548404.

[95] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Bayesian Personalized Ranking with Multi-Channel User Feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 361–364. Association for Computing Machinery, 2016. URL https://doi.org/10.1145/2959100.2959163.

[96] Malte. Ludewig and Dietmar Jannach. Evaluation of Session-Based Recommendation Algorithms. *User Modeling and User-Adapted Interaction*, 28(4-5):331–390, 2018. URL https://doi.org/10.1007/s11257-018-9209-6.

[97] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. Empirical Analysis of Session-based Recommendation Algorithms. *User Modeling and User-Adapted Interaction*, 31(1):149–181, 2021. URL https://doi.org/10.1007/s11257-020-09277-1.

[98] Scott M. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS '17, page 4768–4777. Curran Associates Inc., 2017.

[99] Egil Martinsson. WTTE-RNN : Weibull Time To Event Recurrent Neural Network A Model for Sequential Prediction of time-to-event in the case of discrete or continuous censored data, recurrent events or time-varying covariates. 2017.

[100] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020. URL https://christophm.github.io/interpretable-ml-book/.

[101] Luca Moschella, Valentino Maiorca, Marco Fumero, Antonio Norelli, Francesco Locatello, and Emanuele Rodolà. Relative Representations Enable Zero-shot Latent Space Communication. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SrC-nwieGJ.

[102] Andrew Yan-Tak Ng. Why AI Projects Fail, Part 4: Small Data, 2019. URL https://www.deeplearning.ai/the-batch/why-ai-projects-fail-part-4-small-data/. Last accessed: 2022-10-04.

[103] Caio Nóbrega and Leandro Marinho. Towards Explaining Recommendations through Local Surrogate Models. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1671–1678. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3297280.3297443.

[104] Julius Odili. The Dawn of Metaheuristic Algorithms. *International Journal of Software Engineering and Computer Systems*, 4:49–61, 2018.

[105] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking : Bringing Order to the Web. In *WWW 1999*, 1999.

[106] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, 73:173–180, 2015. URL https://www.sciencedirect.com/science/article/pii/S0950705114003530.

[107] Weike Pan, Qiang Yang, Wanling Cai, Yaofeng Chen, Qing Zhang, Xiaogang Peng, and Zhong Ming. Transfer to Rank for Heterogeneous One-Class Collaborative Filtering. *ACM Transactions on Information Systems*, 37(1), 2019. URL https://doi.org/10.1145/3243652.

[108] Xiaofeng Pan, Ming Li, Jing Zhang, Keren Yu, Hong Wen, Luping Wang, Chengjun Mao, and Bo Cao. MetaCVR: Conversion Rate Prediction via Meta Learning in Small-Scale Recommendation Scenarios. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 2110–2114. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3531733.

[109] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. *ArXiv*, abs/1806.07421, 2018. URL https://api.semanticscholar.org/CorpusID:49324724.

[110] Tu Minh Phuong, Tran Cong Thanh, and Ngo Xuan Bach. Neural Session-Aware Recommendation. *IEEE Access*, pages 86884–86896, 2019.

[111] Maleeha Qazi, Glenn M. Fung, Katie J. Meissner, and Eduardo R. Fontes. An Insurance Recommendation System Using Bayesian Networks. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 274–278. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3109859.3109907.

[112] Maleeha Qazi, Kaya Tollas, Teja Kanchinadam, Joseph Bockhorst, and Glenn Fung. Designing and Deploying Insurance Recommender Systems Using Machine Learning. *WIREs Data Mining and Knowledge Discovery*, 10(4), 2020.

[113] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. BPRH: Bayesian Personalized Ranking for Heterogeneous Implicit Feedback. *Information Sciences*, 453:80–98, 2018. URL https://www.sciencedirect.com/science/article/pii/S0020025516315742.

[114] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 130–137. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3109859.3109896.

[115] Ramin Raziperchikolaei, Guannan Liang, and Young-joo Chung. Shared Neural Item Representations for Completely Cold Start Problem. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pages 422–431. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3460231.3474228.

[116] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461. AUAI Press, 2009. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1630&proceeding_id=25.

[117] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101. Association for Computational Linguistics, 2016. URL https://aclanthology.org/N16-3020.

[118] Claudia Roberts, Ehtsham Elahi, and Ashok Chandrashekar. CLIME: Completeness-Constrained LIME. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 950–958. Association for Computing Machinery, 2023. URL https://doi.org/10.1145/3543873.3587652.

[119] Renata Lopes Rosa, Gisele Maria Schwartz, Wilson Vicente Ruggiero, and Demóstenes Zegarra Rodríguez. A Knowledge-Based Recommendation System That Includes Sentiment Analysis and Deep Learning. *IEEE Transactions on Industrial Informatics*, 15 (4):2124–2135, 2019.

[120] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-Session Modeling for Session-Based Recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, DLRS@RecSys '17, page 24–31. Association for Computing Machinery, 2017. URL https://doi.org/10.1145/3125486.3125491.

[121] Guillaume Salha-Galvan, Romain Hennequin, Benjamin Chapus, Viet-Anh Tran, and Michalis Vazirgiannis. Cold Start Similar Artists Ranking with Gravity-Inspired Graph Autoencoders. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pages 443–452. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3460231.3474252.

[122] Bipin Patwardhan Sanghamitra Mitra, Nilendra Chaudhari. Leveraging Hybrid Recommendation System in Insurance Domain. *International Journal Of Engineering And Computer Science*, 3:8988–8992, 2014.

[123] Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. ProtoCF: Prototypical Collaborative Filtering for Few-shot Recommendation. In *Proceedings of the 15th*

ACM Conference on Recommender Systems, RecSys '21, pages 166–175. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3460231.3474268.

[124] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, 2007.

[125] Tobias Schnabel and Paul N. Bennett. Debiasing Item-to-Item Recommendations With Small Annotated Datasets. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 73–81. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3383313.3412265.

[126] Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. A Review-Aware Graph Contrastive Learning Framework for Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1283–1293. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3531927.

[127] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations*, 2013. URL https://api.semanticscholar.org/CorpusID:1450294.

[128] Viomesh Kumar Singh, Sangeeta Sabharwal, and Goldie Gabrani. Comprehensive Analysis of Multimodal Recommender Systems. In *Data Intelligence and Cognitive Informatics*, pages 887–901. Springer Singapore, 2021.

[129] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: Removing Noise by Adding Noise. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:11695878.

[130] Henrik Stormer. Improving E-Commerce Recommender Systems by the Identification of Seasonal Products. 2007.

[131] Sven Strickroth and Niels Pinkwart. High Quality Recommendations for Small Communities: the Case of a Regional Parent Network. In *Proceedings of the 6th ACM Conference on Recommender Systems*, RecSys '12, pages 107–114. Association for Computing Machinery, 2012. URL https://doi.org/10.1145/2365952.2365976.

[132] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441–1450. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3357384.3357895.

[133] Huimin Sun, Jiajie Xu, Kai Zheng, Pengpeng Zhao, Pingfu Chao, and Xiaofang Zhou. MFNP: A Meta-optimized Model for Few-shot Next POI Recommendation. In Zhi-Hua Zhou, editor, *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, IJCAI '21, pages 3017–3023. ijcai.org, 2021. URL https://doi.org/10.24963/ijcai.2021/415.

[134] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. Multi-Modal Knowledge Graphs for Recommender Systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 1405–1414. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3340531.3411947.

[135] Xuehan Sun, Tianyao Shi, Xiaofeng Gao, Yanrong Kang, and Guihai Chen. FORM: Follow the Online Regularized Meta-Leader for Cold-Start Recommendation. In *Proceedings of the 44th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 1177–1186. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3404835.3462831.

[136] Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2012. URL https://doi.org/10.2200/S00433ED1V01Y201207DMK005.

[137] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 23–32. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3383313.3412489.

[138] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

[139] Fei Tao, Xudong Liu, Xiaorong Mei, Lei Yuan, and Ji Liu. KWAI-AD-AudVis, 2020. URL https://doi.org/10.5281/zenodo.4023390.

[140] Nava Tintarev and Judith Masthoff. *Explaining Recommendations: Design and Evaluation*, pages 353–382. Springer US, 2015. URL https://doi.org/10.1007/978-1-4899-7637-6_10.

[141] Luan Tran, Xiaoming Liu, Jiayu Zhou, and Rong Jin. Missing Modalities Imputation via Cascaded Residual Autoencoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4971–4980, 2017.

[142] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 30Music Listening and Playlists Dataset. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16, 2015*, volume 1441 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015. URL https://ceur-ws.org/Vol-1441/recsys2015_poster13.pdf.

[143] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. DropoutNet: Addressing Cold Start in Recommender Systems. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems*, NeurIPS '17, pages 4957–4966, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/dbd22ba3bd0df8f385bdac3e9f8be207-Abstract.html.

[144] Cheng Wang, Mathias Niepert, and Hui Li. LRMM: Learning to Recommend with Missing Modalities. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3360–3370. Association for Computational Linguistics, 2018. URL https://aclanthology.org/D18-1373.

[145] Qi Wang, Liang Zhan, Paul Thompson, and Jiayu Zhou. Multimodal Learning with Incomplete Modalities by Knowledge Distillation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1828–1838. Association for Computing Machinery, 2020. URL https://doi.org/10.1145/3394486.3403234.

[146] Qinyong Wang, Hongzhi Yin, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. Enhancing Collaborative Filtering with Generative Augmentation. In *Proceedings of the 25th ACM International Conference on Knowledge Discovery and Data Mining*, SIGKDD '19, pages 548–556. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3292500.3330873.

[147] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. Sequential Recommender Systems: Challenges, Progress and Prospects. In *International Joint Conference on Artificial Intelligence*, 2019. URL https://api.semanticscholar.org/CorpusID:199466417.

[148] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. Graph Learning based Recommender Systems: A Review. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, IJCAI '21, pages 4644–4652. ijcai.org, 2021. URL https://doi.org/10.24963/ijcai.2021/630.

[149] Shuai Wang, Kun Zhang, Le Wu, Haiping Ma, Richang Hong, and Meng Wang. Privileged Graph Distillation for Cold Start Recommendation. In *Proceedings of the 44th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 1187–1196. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3404835.3462929.

[150] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM International Conference on Knowledge Discovery & Data Mining*, SIGKDD '17, pages 950–958. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3292500.3330989.

[151] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, page 165–174. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3331184.3331267.

[152] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. MMGCN: Multi-Modal Graph Convolution Network for Personalized Recommendation of Micro-Video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 1437–1445. Association for Computing Machinery, 2019. URL https://doi.org/10.1145/3343031.3351034.

[153] Huizi Wu, Cong Geng, and Hui Fang. Causality and Correlation Graph Modeling for Effective and Explainable Session-Based Recommendation. *ACM Transactions on the Web*, 18(1), 2023. URL https://doi.org/10.1145/3593313.

[154] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised Graph Learning for Recommendation. In *Proceedings of the 44th International ACM Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 726–735. Association for Computing Machinery, 2021. URL https://doi.org/10.1145/3404835.3462862.

[155] Junda Wu, Zhihui Xie, Tong Yu, Handong Zhao, Ruiyi Zhang, and Shuai Li. Dynamics-Aware Adaptation for Reinforcement Learning Based Cross-Domain Interactive Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 290–300. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3531969.

[156] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-Based Recommendation with Graph Neural Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI '19, pages 346–353. AAAI Press, 2019. URL https://doi.org/10.1609/aaai.v33i01.3301346.

[157] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Mengyin Lu, and Liefeng Bo. Multi-Behavior Enhanced Recommendation with Cross-Interaction Collaborative Relation Modeling. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1931–1936, 2021.

[158] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy X. Huang. Hypergraph Contrastive Collaborative Filtering. In *Proceedings of the 45th International*

ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, pages 70–79. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3532058.

[159] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal Recommendation on Graphs via Long- and Short-term Preference Fusion. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery & Data Mining*, SIGKDD '10, pages 723–732. Association for Computing Machinery, 2010. URL https://doi.org/10.1145/1835804.1835896.

[160] Jingcao Xu, Chaokun Wang, Cheng Wu, Yang Song, Kai Zheng, Xiaowei Wang, Changping Wang, Guorui Zhou, and Kun Gai. Multi-Behavior Self-Supervised Learning for Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 496–505. Association for Computing Machinery, 2023. URL https://doi.org/10.1145/3539618.3591734.

[161] Wei Xu, Jiajia Wang, Ziqi Zhao, Caihong Sun, and Jian Ma. A Novel Intelligence Recommendation Model for Insurance Products with Consumer Segmentation. *Journal of Systems Science and Information*, 2:16 – 28, 2014. URL https://doi.org/10.1515/JSSI-2014-0016.

[162] Weilong Yao, Jing He, Guangyan Huang, Jie Cao, and Yanchun Zhang. Personalized Recommendation on Multi-Layer Context Graph. In *Proceedings of the 14th International Conference on Web Information Systems Engineering*, volume 8180 of *WISE '13*, pages 135–148. Springer, 2013. URL https://doi.org/10.1007/978-3-642-41230-1_12.

[163] Zixuan Yi, Xi Wang, Iadh Ounis, and Craig Macdonald. Multi-modal Graph Contrastive Learning for Micro-video Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1807–1811. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3532027.

[164] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential Recommender System based on Hierarchical Attention Networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI '18, pages 3926–3932. ijcai.org, 2018. URL https://doi.org/10.24963/ijcai.2018/546.

[165] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized Entity Recommendation: a Heterogeneous Information Network Approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 283–292. Association for Computing Machinery, 2014. URL https://doi.org/10.1145/2556195.2556259.

[166] Yongfeng Zhang and Xu Chen. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends in Information Retrieval*, 14(1):1–101, 2020. URL https://doi.org/10.1561/1500000066.

[167] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H. Chi. Improving user topic interest profiles by behavior factorization. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1406–1416. International World Wide Web Conferences Steering Committee, 2015. URL https://doi.org/10.1145/2736277.2741656.

[168] Jiayin Zheng, Juanyun Mai, and Yanlong Wen. Explainable Session-Based Recommendation with Meta-Path Guided Instances and Self-Attention Mechanism. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2555–2559. Association for Computing Machinery, 2022. URL https://doi.org/10.1145/3477495.3531895.

[169] Feng Zhu. Douban Dataset (Ratings, Item Details, User Profiles, Tags, and Reviews), 2021.