5-17-2022

Semantic Embedded Sequential Recommendation for E-Commerce Products through Mining Customers’ Historical Interactions and Products’ Data

Mahreen Nasir Butt
University of Windsor

Semantic Embedded Sequential Recommendation for E-Commerce Products through Mining Customers' Historical Interactions and Products' Data

By

Mahreen Nasir Butt

A Dissertation
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2022

Semantic Embedded Sequential Recommendation for E-Commerce Products through Mining Customers' Historical Interactions and Products' Meta Data

by

Mahreen Nasir Butt

APPROVED BY:

29 April 2022

Declaration of Co-Authorship and Previous Publications

I. Co-authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows: Chapter 6 briefly refers to the outcomes of publications co-authored with the supervisor along with other undergraduate and graduate students at the WODD Lab, School of Computer Science, University of Windsor. The details regarding contributions made by each author towards the publications are provided as follows:

- For the publication titled "Improving e-commerce product recommendation using semantic context and sequential historical purchases" co-authored with Ezeife, C. I. Gidado, A., the key idea, main contributions, experimental designs, data analysis, interpretation, and writing were performed by myself under the supervision of Ezeife, C. I.
- For the publication titled "Semantic Enhanced Markov Model for Sequential E-commerce Product Recommendation" co-authored with Ezeife, C. I., the key idea, main contributions, experimental designs, data analysis, interpretation, and writing were performed by myself under the supervision of Ezeife, C. I.
- For the publication "Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation" co-authored with Ezeife, C. I., the key idea, main contributions, experimental designs, data analysis, interpretation, and writing were performed by myself under the supervision of Ezeife, C. I.

- For the publication titled "Semantics Embedded Sequential Recommendation for E-Commerce Products (SEMSRec)", co-authored with Ezeife, C. I., the key idea, main contributions, experimental designs, data analysis, interpretation, and writing were performed by myself under the supervision of Ezeife, C. I.

- For publication titled “The HSPRec E-Commerce System Open Source Code Implementation”, co-authored with Ezeife, C. I., Chaturvedi, R., and Veliz Castro, A., primary contributions are attributed to Ezeife, C. I, in terms of the key idea, experimental designs, data analysis, interpretation, writing and feedback on refinement; I provided assistance in experimentation and analysis, contributed in providing feedback on refinement of presentation of ideas and editing of the manuscript; Chaturvedi, R assisted in manuscript editing and formatting for submission to the conference and Veliz Castro, A., executed the experiments and initial analysis.

- For the publication titled “Extracting High Profit Sequential Feature Groups of Products using High Utility Sequential Pattern Mining” co-authored with Motwani, Priyanka and Ezeife, C. I., the key idea, main contributions in experiments and analysis, manuscript writing were executed by the student Motwani, Priyanka under the supervision of Ezeife, C. I.; I contributed in providing feedback on refinement of the manuscript in terms of formating and editing.

- For the publication titled “Mining sequential patterns of historical purchases for e-commerce recommendation” co-authored with Bhatta, R. and Ezeife, C. I, the key idea, main contributions in experiments and analysis, manuscript writing were executed by the student Bhatta, R., under the supervision of Ezeife, C. I.; I provided assistance in reviewing the experimental results and contributed in providing feedback during the preparation of the manuscript and presentation of ideas.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publications

This thesis includes seven papers with four original papers (three Journal and one Conference) stemming directly from this thesis and three conference papers (that resulted from work as co-author with the supervisor and other graduate students at the WODD Lab, School of Computer Science, University of Windsor). Details of publications along with their status are as follows:

	Publication Title	Status
Chapter 5, 6 and 8	Nasir, M., Ezeife, C. I. & Gidado, A (2021). Improving e-commerce product recommendation using semantic context and sequential historical purchases. Springer's International Journal of Social Network Analysis Mining, 11, 82. https://doi.org/10.1007/s13278-021-00784-6	Published
Chapter 6, 7 and 8	Nasir, M., Ezeife, C. I. Semantic Enhanced Markov Model for Sequential Ecommerce Product Recommendation. International Journal of Data Science and Analytics. Springer Nature.	Accepted
Chapter 1 and 2	Nasir, M., Ezeife, C. I. A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation. SN Computer Science. Springer Nature.	Accepted with Revision (under review)
Chapter 4, 6 and 8	Nasir, M., & Ezeife, C. I. (2020, December). Semantics Embedded Sequential Recommendation for E-Commerce Products (SEMSRec). In 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 270-274). IEEE.	Published

Publications as co-author and related to thesis are as follows:

	Publication Title	Status
	Ezeife, C. I., Nasir, M., Chaturvedi, R., Veliz Castro, A. (2021, August). The HSPRec E-Commerce System Open Source Code Implementation. In IEEE/ACIS 21st International Fall Conference on Computer and Information Science (ICIS 2021-Fall) in Xi'an, China	Published
	Motwani, Priyanka., Ezeife, C. I., Nasir, M., (2021, September). Extracting High Profit Sequential Feature Groups of Products using High Utility Sequential Pattern Mining. In International Conference on Advanced Data Mining and Applications (ADMA) in Sydney, Australia.	Published
	Bhatta, R., Ezeife, C. I., & Nasir, M. (2019, August). Mining sequential patterns of historical purchases for e-commerce recommendation. In International Conference on Big Data Analytics and Knowledge Discovery (pp. 57-72). Springer, Cham.	Published

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's

Abstract

E-commerce Recommendation Systems (ERS) facilitate customers' purchase decision by recommending products or services of interest. Designing a recommender system targeted towards an individual customer's need is crucial for retailers to increase revenue and retain customers' loyalty. Collaborative Filtering (CF), a common recommendation technique, takes user-item interaction matrix as input which represents interactions either explicitly (users ratings) or implicitly (users' browsing or buying behavior) and outputs top item recommendations for each target user, by finding similarities among users or items. The input matrix suffers from (i) sparsity (has low user item interactions), (ii) cold start (an item cannot be recommended if no ratings exist). Content Based method, on the other hand, generates recommendations based on the content (features) of the item and suffers from content overspecialization (lack of diversity in recommended products) due to the use of specific features only.

Furthermore users' interests and preferences change with time. The time stamp of a user interaction (click or purchase event) is an important characteristic. Learning the sequential patterns of user interactions based on the timestamps are useful to understand their long and short term preferences and predict the next items for recommendation. Sequential Pattern Mining mines frequent or high utility sequential patterns from a sequential database comprising of historical purchase or click sequences. Conventional recommendation systems (ChoiRec12, SuChen15, HPCRec18, HSPRec19) utilize mining techniques such as clustering, frequent and sequential pattern mining along with click and purchase similarity measures for next item recommendation. However, the performance of these systems is still limited when the matrix is sparse, as the number of items keep increasing rapidly. Additionally, models utilizing sequential pattern mining suffer from (i) lack of personalization:

patterns are not targeted for a specific customer, as they infer decisions based on a global view of sequences and (ii) lack of contextual similarities among recommended items: they can only recommend items that exist as a result of a matching rule generated from frequent sequential purchase pattern(s).

To better understand users' preferences and to infer the inherent meaning of the items (e.g., context in which they are used), this thesis, explores the effectiveness of utilizing semantic knowledge (meaningful relationships between items) extracted from items' meta data (title, description and brand) and customers' purchase histories to compute semantic similarities between items according to their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products based on their characteristics. The extracted semantic knowledge is then integrated into different phases of recommendation process such as (i) pre-processing, to learn relationships between items, (ii) candidate item generation and (iii) generating semantic rich and sequential next item recommendations. During the candidate item generation phase, techniques developed include (a) mining semantic rich sequential patterns, (b) enriching the item matrix in Collaborative Filtering to select Top-N candidates that show semantic and sequential relationships between items and (c) enhancing the Transition Probability Matrix in the Markov Model method. The third phase of generating semantic rich sequential recommendations is accomplished by using semantic rich (a) Sequential Pattern Rules, (b) item based Collaborative Filtering or (c) Markov Model depending upon the method used to generate candidate items. Thus, the inclusion of semantic knowledge into all phases of recommendation process can address the issues of sparsity, coldstart, content overspecialization and provide recommendations which are diverse, similar in context and better reflect user's long and short term interests. Experimental results on publicly available e-commerce data sets such as Amazon and Online Retail has shown that the proposed model has improved performance over existing systems.

Keywords: Recommendation Systems, Sequential Pattern Mining, Semantics, Click-stream data, Historical Purchases, RecSys, Collaborative Filtering, TF-IDF, Vector Space Model, Cold Start, Sparsity, E-commerce

Dedication

Dedicated to my parents and my brother for their unconditional support, love, motivation, wisdom and prayers which enabled me to complete this journey and fulfil my dreams.

(1) In the name of Allah, the Beneficient, the Merciful.

(2) Praise be to Allah, Lord of the Worlds,

(3) the most Gracious, the most Merciful.

(4) The Master of the day of Judgement.

(5) You alone we worhsip and You alone we ask for help.

(6) Guide us to the straight path,

(7) the path of those upon whom You have bestowed favour, not of those who earned Your anger, nor of those who went astray.

(Quran 1:1-7)

Acknowledgements

The journey of a thousand miles begin with one step ˜(Lau Tzu)

Today when my journey towards Ph.D. is at the finish line, my mind is flushed with memories of each and every step that I undertook before and during this journey.

After Allah's countless blessings upon me, I could not thank enough my parents and my brother who are my role models, my source of inspiration and motivation. Their optimism, unconditional support and confidence in me enabled me where I am today. Their wisdom and encouragement has always been a beacon of light for me whenever I found myself stuck in a dark tunnel seeing no light at the end. Being a first generation female student, reaching to university level and graduating with a Doctor of Philosophy degree from a foreign university as an international student brought a lot of challenges. During this time, at certain points, I had to make hard choices to stay firm on my decision and I am truly grateful to my family for supporting me in all my decisions and giving me the freedom to fill the canvas of my life with my choice of colors.

I would also like to thank my professors at University of Windsor, my thesis supervisory committee members, Dr. Dennis Borisov, Dr. Boubaker Boufama and Dr. Sherif Saad for their valuable feedback, co-operation and continuous support throughout. I am much grateful to my thesis supervisor, Dr. Christie Ezeife for her continuous support, thorough feedback on my submissions and her timely response despite her busy schedules. I want to thank her for all her efforts and pushing me to my limits. Her valuable advice at every stage helped and guided me in shaping up this thesis. Thanks to Dr. Ezeife as well for all the research assistantship financial support through her NSERC and other grants.

I also want to thank my friends and colleagues at the University of Windsor for their support and best wishes.

I also acknowledge that this research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an operating grant (OGP-0194134) and a University of Windsor grant received by Dr. Christie Ezeife, an Ontario Graduate Scholarship and Doctoral Entrance Scholarship.

Last but not the least, thank you to each and everyone at University of Windsor for being a part of this journey.

I would like to end by sharing a stanza from one of my favourite poems “The Road Not Taken” by Robert Frost, which pretty much describes my emotions towards this journey.

Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CB	Content Based
CF	Collaborative Filtering
MM	Markov Model
PS	Purchase Sequences
RS	Recommendation Systems
SRecSys	Sequential Recommendation System
SPM	Sequential Pattern Mining
SID	Sequence ID
SKE	Semantic Knowledge Extraction
SKI_SPM	Semantic Knowledge Integration into Sequential Pattern Mining
SKI_CF	Semantic Knowledge Integration into Collaborative Filtering
SKI_MM	Semantic Knowledge Integration into Markov Model
SPP	Sequential Purchase Patterns
SSR_SPM	Semantic and Sequential Recommendation using Sequential Pattern Mining
SSIR_CF	Semantic and Sequential Knowledge Integration and Recommendation using Collaborative Filtering
SSR_MM	Semantic and Sequential Recommendation using Markov Models
TID	Transaction ID

Chapter 1

Introduction

This chapter is an introduction to Recommendation Systems (RS) with e-commerce product recommendation as application domain, its components (input phase, candidate generation and output phases). It will also highlight the motivation behind designing an RS that leverages the use of sequential and semantic associations between items and users into all phases of an RS for the purpose of generating useful recommendations. It will also describe the main contributions.

1.1 Recommendation Systems

1.1.1 E-commerce Recommendation Systems

A Recommendation System (RS) provides suggestions to users according to their interests (Aggarwal, 2016, Ricci et al., 2015). For example, many companies and service providers such as Amazon[1], Facebook[2], Netflix[3]and career builders[4] are providing recommendations to users for products, friends, movies and jobs respectively (Schafer et al., 2001). E-Commerce Recommendation systems assist customers in decision making by recommending items that are likely to be in their interest (Jannach et al., 2010). They facilitate the cus-

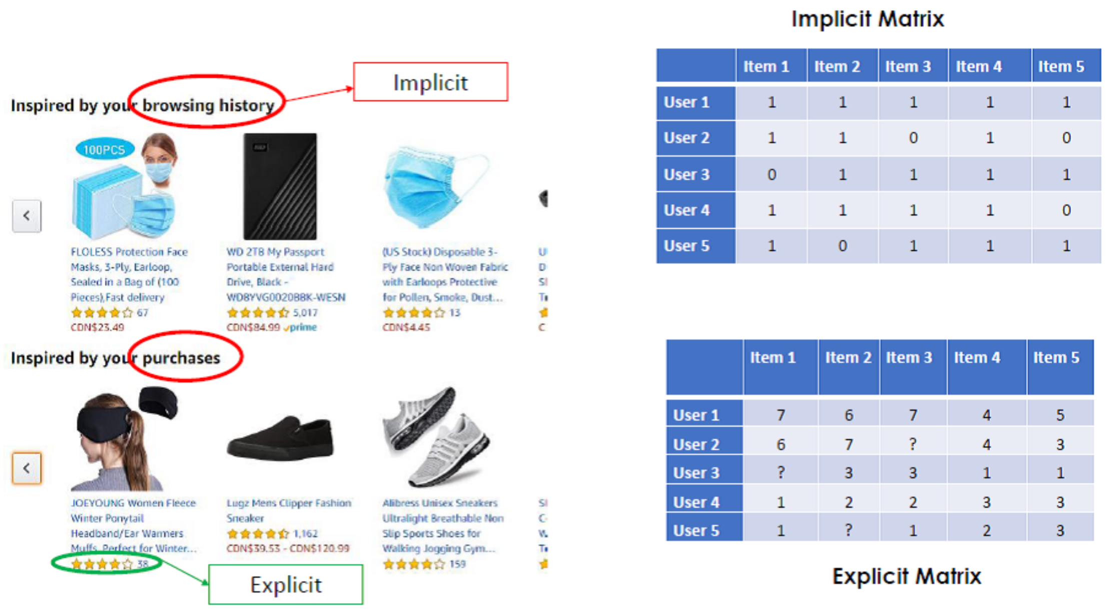
[1]https://www.amazon.com/
[2]https://www.facebook.com/
[3]https://www.netflix.com/ca/
[4]https://www.careerbuilder.ca/

tomers' search process and narrow down the choices out of thousands of available products to save customers' time. Top technology e-commerce companies (e.g., Amazon, Alibaba) deploy customized recommendation systems to cater to the needs of their growing customers and maintain their market share.

One of the most widely-used recommendation technique is *Collaborative Filtering (CF)* which recommend users' products that they might be interested in based on the taste of other similar users (Schafer et al., 2007). In CF, customer's interest in a product is saved in a user-item rating matrix. For example, we have a list of m users $U = \{u_1, u_2, \dots\dots\dots, u_m\}$ and n items $P = \{p_1, p_2, \dots, p_n\}$. The system constructs an $m \times n$ user-item matrix, where each entry $r_{i,j}$ denotes user interactions (clicks, purchases, views) u_i for an item p_j, CF algorithm then either predicts a rating for an item or recommends a list of most likable Top-K items for user u_i. The rating matrix can be (a) *implicit*- in which ratings are collected based on clickstream or historical purchase data such as tracking users' navigational behavior on a site or (b) *explicit* where customers provide ratings, for example, users rating movies on a scale of 1 to 5. Figure 1.1 shows an illustration of an implicit and explicit rating matrix created from user's browsing and reviews' history from an e-commerce site. The entries "1" or "0" in the implicit matrix shows that the user has interacted with those items and similarly an explicit rating provided by the customer for an item is recorded in the explicit matrix. A "0" or a "?" indicates absence of users' interaction with that particular item.

Content based on the other hand generate recommendations based on the content (features) of the item (Adomavicius and Tuzhilin, 2011). The system creates user profiles based on item features (key words) extracted from items which the user has consumed in the past. Different models (e.g., TF-IDF) are then used to compute similarity (relationship) between already consumed items and the items available in the catalog. Items with higher similarity score are then recommended to the customer.

While Collaborative Filtering based recommendation does not consider items' properties and only uses user interactions with items, it suffers from some limitations such as (1) sparse rating matrix- as users only rate a small number of items out of tens of thousands of available products and (2) cold start- new item cannot be recommended initially when it is introduced to a Collaborative Filtering system with no ratings. Alternately, content based approaches suffer from (1) content overspecialization (lack of diversity in recommended products) due



Implicit Matrix

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	1	1	1	1
User 2	1	1	0	1	0
User 3	0	1	1	1	1
User 4	1	1	1	1	0
User 5	1	0	1	1	1

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	7	6	7	4	5
User 2	6	7	?	4	3
User 3	?	3	3	1	1
User 4	1	2	2	3	3
User 5	1	?	1	2	3

Explicit Matrix

FIGURE 1.1: Implicit and explicit user-item rating in CF based recommendation systems

to the use of specific item features only (Adomavicius and Tuzhilin, 2011).

Hybrid recommendations systems were introduced to address the limitations of both CF and content based systems. The idea is to generate recommendations by considering user preferences (implicit or explicit) on items along with the content (product features).

1.1.2 Challenges in E-Commerce Recommendation Systems

E-commerce Recommendation Systems face certain challenges including:

1. **Cold Start** problem- In cold start problem items cannot be recommended due to the lack of information about the user and/or the item. Cold-start problem presents a collective issue of new item and new user to RSs. There are two types of cold start (i) New Item: A new item cannot be recommended initially when it is introduced to a Collaborative Filtering system with no ratings. For instance, MovieLens (movielens.org) cannot recommend new movies until these have got some initial ratings. (ii) New User: When a new user enters the system, it is bit hard to find similar users or to create a content based profile as previous preferences (such as browsing history, likes/dislikes) of user are not available.

2. **Sparsity** – Sparsity occurs when majority of the users do not rate most of the items and consequently the ratings matrix becomes very sparse which limits the chances of finding a set of users with similar ratings. In other words, this problem arises when there are many items to be recommended but only few recommendations are provided (because of fewer available ratings on items).

3. **Lack of Sequential information in Recommendation**

   These traditional recommender systems (Collaborative Filtering and Content based methods) model users' interactions with items (e.g., clicks, add to cart, views or purchases) in a static way and can only capture users' general preferences without considering the sequential behaviour of users'-item interactions. As users' interests and preferences change with time, learning the sequential behaviour in users' interactions based on the timestamps are useful to (i) understand their long and short term preferences and (ii) predict the next items for purchase based on finding the sequential dependency of an event (e.g., click or purchase) on its preceding event as the time interval between any such interactions provide useful insights about users' behavior. So, a Sequential Recommendation System views the interactions of a user as a sequence and aims to predict which item the user will interact with next. Hence, the problem of Sequential Recommendation System can be stated as: Given past item interactions (e.g., clicks, views, purchases) of a user, the goal of a Sequential Recommendation System is to predict the future item(s) that are of interest to the user.

1.1.3 How E-commerce Recommendation is Different from Recommendation in other Domains

1. **Increased Number of Products**

   Retailers and vendors keep on adding new products to their catalog to attract customers. As this addition in products is good to provide more choices to customers, at the same time it brings the challenge of sparsity (as customers do not interact with all products out of millions of available products).

2. **Long and Short Term User Preferences**

   Users browsing and purchasing preferences change with time. For example, one of the scenarios can be where user has a long term preference towards a particular dairy product preferences, but recently due to certain dietary restrictions, they shift towards other brands/dairy products. Another scenario could be change in user's browsing and purchase preferences due to Covid'19 pandemic. For example, we can see high demand in purchase of sanitizing products, masks and other such products. So, monitoring these static (long term) preferences (e.g., a particular clothing brand/style) and dynamic short term intent(e.g., purchase of sanitizing products) are important to generate tailored recommendations to customers.

3. **Regular Buyers and Temporary Buyers**

   It is both important and challenging to differentiate between regular and temporary customers. The target should be to retain regular customers while at the same time convert the temporary customers to regular customers (e.g., by offering them incentives, cash back offers etc). For example, we have seen in recent times during the Covid'19 pandemic, due to the forced lockdowns people turned to online shopping even if they never preferred it. However, after the ease in lockdown restrictions, these customers might prefer in-store shopping from their regular store of choice.

4. **Huge Diversity across Various Product Categories**

   Big e-commerce platforms like Amazon, Alibaba have a diverse collection of product categories ranging from digital devices, books, video games, electronics, home décor, furniture, office supplies, grocery, health and beauty, clothing sports to name a few. These product categories are then further sub classified into several categories which creates huge diversity among all available products.

5. **Availability of Substitute Products**

   Due to ever increasing number of products, a lot of substitutes exists for products. For example, in case of house hold (e.g., kitchen items), if the user is looking for a non-stick cook-ware set, there will be lots of substitutes available (e.g., similar products from other brands). So, if the customer is not looking for product from a specific brand, she may likely to choose from the shown substitute products. Additionally, cook-wares with other specifications (e.g., ceramic) can also be available as a substitute.

1.2 Components of a Recommendation Framework

A recommender system aims at recommending items to users that match their interest. Recommender systems work by creating a user profile which represents a user's preference on items she has interacted with or consumed in the past. This user profile (input) can be created in a number of ways such as (i) explicit- by observing users' behavior in terms of ratings or reviews they provided on items or (ii) implicit- by monitoring customers' buying or browsing behavior. Other auxiliary information such as user demographics or item attributes (content) can also be used to enrich the user profile for better recommendations. For example, to capture characteristics of an item, an item profile can be created either based on its features (descriptive) such as movie genre, actor, and director information or in the form of numeric ratings (quantitative) given to an item by the user. Once the user and item profile are available, the recommendation system aims to find candidate items for recommendation to the target user. A typical recommendation system framework consists of three phases which are i) input, ii) candidate generation and iii) output (recommendation)(Aggarwal, 2016).

1.2.1 Input Phase (Data Sources)

This phase records user-item interactions (explicit or implicit). This facilitates the process of creating user and item profile which are then used in the next phase during the process of generating candidates, that is, filtering items that will be of interest to the user. The quality of the input (information sources) can significantly improve the selection of suitable candidates. Various types of information sources are available including (traditional and auxiliary) which are used as an input to the recommendation framework to enrich the recommendation process. In the next subsections, we will discuss about these types. Section 2.1 provides a classification of RS on the basis of input data sources used.

Traditional Data Sources- Traditional information sources consist of gathering users' interest on items by monitoring their implicit or explicit behaviors where ***Implicit*** actions are captured by inferring how a user responds to an item. For example, on an E-commerce site, which items a user has browsed, clicked, added to his favourite items' list, added to cart or purchased. These implicit actions indicate a user's interest on an item and are helpful in determining their future preferences. One way of expressing implicit behaviors is by storing binary values in the user item rating matrix where a “1” indicates behavior on an item such as clicked or purchased and a “0” otherwise. A disadvantage is that it cannot fully reflect users’ preference on the item as a value of “0” does not necessarily mean that a user is not interested in the item. Since there are millions of products, it is very likely possible that the user is unaware of the existence of such items. ***Explicit*** behaviors on the other hand are recorded when users’ explicitly provide feedback on the item. A common example is rating an item on a five star scale where “1” is the lowest rating and “5” is the highest. Another way of extracting users’ explicit feedback is from text reviews. In this case, if the review is positive, a value of “1” is stored in the user-item matrix, “0” for a neutral review and “-1” for a negative feedback. A common example is extracting user’s preferences from the reviews they provide after consuming a product.

Auxiliary Input Sources- These information sources complement the traditional information sources and can better reflect user’s interest by capturing more insights about users’ behavior. These information can be classified as (a) user and item information (e.g., meta data), (b) information contributed by users (e.g., tags, geotags, multimedia content, free comments and reviews) and (c) information associated with user-item interaction also known as context. An example of such information can be when a user is interacting with an item, such as purchasing an item, watching a movie or listening to a song (Adomavicius and Tuzhilin, 2011).
Consider Table 1.1 showing a sample of customers’ historical purchase records containing information about the purchase (Invoice No.), products purchased (Stock Code), date and time of purchase (Invoice Date), and customer id who made the purchase (Customer ID). For example, Customer with ID 17850 purchased products “20674” and “21242” on January 12th, 2010 at 8:26 am. Additionally, Table 1.2 shows product meta data (description of purchased products along with other product attributes such as title and brand information).

1.2.2 Candidate Generation Phase

This step involves the creation of user and item profiles from the user-item interactions and other auxiliary information sources obtained from the input phase. The processing in this

Table 1.1: Sample historical product purchase record of customers

Invoice No.	Stock Code	Invoice Date	Customer ID
536365	20674	12/1/10 8:26	17850
536365	21242	12/1/10 8:26	17850
536365	20675	14/1/10 9:26	17851
536365	20675	15/1/10 11:28	17852
536365	20674	12/1/10 11:28	17852
536365	21242	12/1/10 11:30	17852
536365	21242	12/1/10 11:30	17852

Table 1.2: Sample product meta data

Stock Code	Title	Description	Brand
20674	Green polka Dot bowl	Earthenware, largest measures 5.5 inch h x 12 inch l x 11.25 inch hand wash	Tag limited
21242	Red retrospot Plate	These beautiful plates are composed of high-rated heavyweight plastic materials rendering the plates leak-free, soak resistant, cut proof and unbreakable.	Silver Spoons
20675	Blue Polka Dot bowl	This polka dot bowl is fun and festive and perfect for that bowl of cereal in the morning or bowl of ice cream in the evening. It is finished in a blue celadon glaze with a sprinkling of matte black polkadots. Dish washer safe.	Creative innovations

step includes generating the set of candidate(s), that is, set of (i) users (neighbors) having similar interests to the target user or (ii) items which are similar in characteristics to the items the user has already interacted with. Various techniques can be used at this stage for the process of generating candidates. Section 2.2 provides classification of RS according to various techniques used for candidate generation by finding associations between items and users such as Similarity Based Associations (Content based and Neighborhood based), Sequential Associations (Traditional Approaches, Model Based and Neural Network Based) and Semantics Based Associations (Top down and Bottom up approaches) (Sect. 2.2).

1.2.3 Output Phase (Recommendation)

In this phase, final recommendations for items are provided to the target user based on the output from the candidate generation phase. The output of the recommendation task can be formulated as rating or a ranking problem.

(a) Recommendation as Rating Problem: In this scenario, the process of recommendation is formulated as a matrix completion task where the goal is to predict the missing ratings in the user-item rating matrix to infer user's preference for unseen items (unknown item ratings). A higher predicted rating for an item will show that user is most likely to show interest in the item. The observed entries in the user-item rating matrix are used as training data to predict the missing ratings for users. However, in most cases, it is not necessary to predict a rating for an item before making recommendation. In this setting, the input to the system will be the sparse user item matrix and the output will be user's predicted ratings for the unseen items.

(b) Recommendation as Ranking Problem: In this setting, instead of predicting missing ratings on unseen items for the target user, the goal is to find top-n users or items (neighbors) similar to target user's preference. This is also referred to as top-n recommendation where the top-n items similar to the target user's interests are recommended. Here, the input to the system will be the sparse user item matrix (with missing ratings) and the output will consist of the set of top-n items that are likely to be of interest to the user. To compute the set of similar (neighbor) users or item, different similarity measures are used such as cosine similarity or pearson correlation. However, in some settings top-n recommendations are generated by first predicting ratings on unseen items and then recommending list of top-n items with high ratings to the target user.

1.3 Motivation

1.3.1 Why need a Sequential Recommendation System

Sequential Recommendation Systems suggest items which may be of interest to a user by modelling the sequential dependencies over the user-item interactions (e.g., clicking, viewing or purchasing items on an online shopping platform) in a sequence (Wang et al., 2021). Learning sequential dependencies between items for next item recommendation assist in modeling user preferences by reflecting the sequential dependency of an event (e.g., click or purchase) on its preceding event. A Sequential Recommender views user interactions as a sequence and predicts the next items with which the user will interact. Items with which a user interacts (e.g., clicked, rated or purchased) can provide a strong indication of her interests and facilitate in learning a good user profile leading to recommendations that match her interests. However, users' interests and preferences change with time. The time stamp of a user interaction (click or purchase event) is an important attribute and learning the sequential patterns of user interactions based on the timestamps are useful to (i) understand the long and short term preferences of user and (ii) predict the next items for purchase by users as the time interval between any such interactions provide useful insights about users' behavior.

In the real world, user-item interactions (e.g., shopping behaviours) are mostly sequentially dependent. For example, in Figure 1.2, we can see the sequential dependencies between purchased items by a user Smith. After Smith has purchased a camera, memory card and a camera case, what item he will purchase next?. Such kind of sequential dependencies commonly exist in transaction data but traditional Collaborative filtering and Content Based recommender systems are unable to capture these sequential dependencies which necessitates the need for developing Sequential Recommendation Systems.

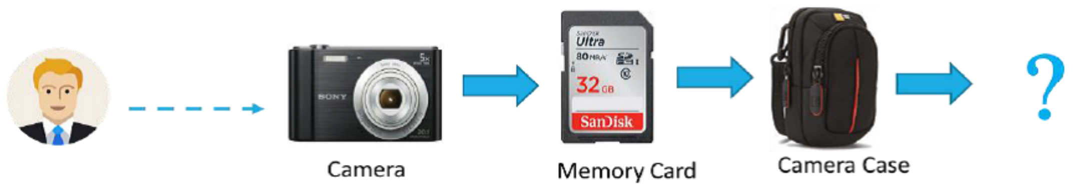


Figure 1.2: Example of a Sequential Recommendation System: After Smith has purchased a camera, memory card and a camera case, what item he will purchase next?

Table 1.3: A sequence database (SDB)

SID	Sequences
1	{a,b},{c},{f,g},{g},{e}
2	{a,d},{c},{b},{a.b,e,f}
3	{a},{b},{f,g},{e}
4	{b},{f,g}

1.3.2 Extracting Sequential Patterns from user interactions using Sequential Pattern Mining

Sequential Pattern Mining mines frequent or high utility sequential patterns from a sequential database such as historical purchase or click sequence database of customers. The problem of Sequential Pattern Mining can be stated as: Given (i) a set of sequential records (called sequences) representing a sequential database $SDB = \{s_1, s_2, s_3, \ldots\ldots\ldots, s_n\}$ with sequence identifiers $1, 2, 3, \ldots\ldots, n$, (ii) a minimum support threshold called min sup ξ and (iii) a set of k unique items or events $I = \{i_1, i_2, \ldots\ldots, i_k\}$, the problem of mining sequential patterns is to find the set of all frequent sub-sequences S in the given sequence database SDB of items I at the given min sup ξ, that are interesting for the user. For example, consider Table 1.3 showing a sequence database containing sequences representing five transactions made by a customer at a retail store. Each single letter in a sequence represents an item. Items between curly brackets represent an itemset. The sequence with SID 1 indicates that a customer purchased items a and b at the same time, then bought item c, then purchased items f and g at the same time, then purchased item g, and finally purchased item e. So, the goal of sequential pattern mining is to find frequent sequence items based on a minimum support threshold. A sequence s is said to be a frequent sequence or a sequential pattern if and only if sup(s) $>=$ minsup, e.g., in this example some of the frequent sequences are $< a >$, $< a, g >$, $< a, c >$, $< a, f >$ and so on. These sequences are frequent as they meet the minimum support (user defines) which is set to 2. So, sequence $< a >$ is frequent as it occurs in three sequences in the sequence database, that is, in sequences with SID 1, 2, and 3. Chapter 2 sec. 2.2.2.1.2 provides algorithmic details on extracting sequential patterns through a sequential pattern mining algorithm such as PrefixSpan (Jian Pei et al., 2001) using a walk through example. Many Sequential Pattern Mining algorithms were introduced in the literature including GSP (Agrawal and Srikant, 1995), SPAM (Ayres et al., 2002), Prefix Span (Jian Pei et al., 2001) and PLWAP (Ezeife et al., 2005).

So, in this thesis, we want to further exploit the effectiveness of sequential relationships between (a)items and (b) users' various interactions by extracting complex sequential patterns of customer clicks and/or purchases and then integrating those learned patterns into the Collaborative Filtering's user-item rating matrix to address its limitations such as (i) cold start- when no ratings are available for a user/item and (ii) sparsity- when there is less user-interaction data. The frequent sequential purchase patterns derived from customers' historical data, can lead to better next item recommendations for the target user by capturing users' short and long term preferences and can improve the accuracy and diversity of recommendations by finding the sequential relationship between frequently purchased items.

1.3.3 Why use Semantics in Sequential Recommendation Systems?

Semantics (meanings) are required to have a deep comprehension of the information conveyed by the textual content and to achieve the goal of improving the quality of user profiles and the effectiveness of intelligent information access platforms. Semantic Aware recommenders (de Gemmis et al., 2015) infer the semantic (meaning) of the words by exploring meanigfull relationships between items through the use of concepts extracted either from knowledge sources (ontologies, encyclopedic resources) or distributional hypothesis (context in which the words are used). It uses implicit semantic representation of item and user profiles (based on the usage of words).

Item profiles created from key words extracted from item descriptions or meta data associated with the item are insufficient to infer user's interests. Content is required to extend and improve user modeling (preferences) and more importantly to address the limitations of Collaborative Filtering systems such as (i) sparsity- when there is less user-interaction data, (ii) cold start- when no ratings are available for a user/item and (iii) lack of transparency and poor recommendation explanation – it is not clear why a particular item such as a movie or a product is recommended to the user. Content extracted from textual features (item descriptions) to learn key-word based user profiles, limits the performance of recommendation systems due to the ambiguities in the natural language representations. Some of them include (i) polysemy – when multiple meanings are associated with the same word such as Turkey (represents a food and a country), (ii) synonymy- when different words represent the same meanings, (iii) multi-word concepts- which is a collection of two or more words whose meanings are interpreted separately when they are used independently, for example, New York refers to a city in USA, but if used individually both terms yield a different meaning, or for example, “hot dog” which when used together refers to steamed sausage sandwich in the context of eating whereas the individual words refer to completely different meanings and (iv) entity named recognition or entity identification- the difficulty to locate and classify elements in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, etc.

Furthermore, while generating candidate item(s) by mining the frequent sequential patterns for next item recommendation, Sequential Pattern Mining techniques suffer from some draw backs which include (i) absence of contextual similarities among recommended products, that is, they recommend items based on a match with the sequential rules generated from sequential patterns and do not consider relationships between items according to their context. For example, if there are two sequential rules for chocolate purchase such as:
(a)Ferrero Rocher, Ferrero Rondnoir $\rightarrow$ Rafeallo and
(b) Kinder Chocolate Bar, Kinder Surprise $\rightarrow$ Kinder Beuno,

where rule (a) (indicating that a purchase of Ferrero Rocher, Ferrero Rondnoir will most likely lead to the purchase of Rafeallo), so for a new purchase sequence where a customer bought Nutella and Rafeallo, he can be recommended to purchase “Ferrero Collection” as they all belong to same brand and have “similar” characteristics such as their ingredients, however if there is no rule for the product “Nutella”, the customer will not be recommended with a product. Conventional sequential rules fail to capture such semantic relationships between products and will only recommend products that exist as a result of a matching rule with products “Nutella and Rafeallo”. Therefore, it cannot recommend products that are similar in semantics as they are based only on the frequent sequential occurrence. i.e., frequency count based on a minimum support threshold. Additionally, they (ii) lack personalization, rules are not targeted for a specific customer, as they infer decisions based on a global view of sequences.

Therefore, semantics (meanings) are required to have a deep comprehension of the information conveyed by the textual content and to achieve the goal of improving the quality of user profiles. More specifically, semantics are needed to (i) better understand the associations between items and therefore recommend diverse items, (ii) avoid typical issues of natural language representations (polysemy, synonymy, multi-word concepts, etc.), (iii) model user preferences in an effective way and (iv) better understand the information spread on social media.

So, in this thesis, we want to further enrich the recommendation process in terms of accuracy by exploring semantic relationships between items according to their (i) usage (e.g., products co-purchased or co-reviewed) and their (ii) textual features by finding similarity between products based on their characteristics and then integrating this semantic knowledge into the sequential recommendation process for generating candidate items (such as while mining sequential patterns), finding Top-N neighbors (such as in Collaborative Filtering) and in Transition Probability Matrix of Markov Model to address the limitations of (i) content overspecialization (lack of diversity in recommendations), (ii) lack of personalization, (iii) sparsity, (iv) cold start and (v) unambiguous prediction problem (when multiple items have same probabilities of purchase after the current purchase). The learnt semantic relationships between items extracted from customers’ historical data and products’ meta data can better represent users’ preferences and hence can provide diverse and tailored recommendations.

1.4 Thesis Contributions

The contributions of this study are to improve the performance of recommendation system in terms of accuracy in the context of sparse user-item interactions by integrating semantic knowledge (relationship) of items extracted from customers' purchase histories based on items' co-occurrence from customers' purchase histories and items' meta data into the candidate generation and recommendation phases of a sequential e-commerce recommendation system. The main contributions of this study are divided into feature and procedural contributions as:

1.4.1 Feature Contributions

The main features proposed in this study are to (i) extract semantic knowledge of items from items' meta data (title, description and brand) and customers' purchase histories to compute semantic similarities between items according to their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products based on their characteristics, (ii) integrate items' semantic knowledge to generate candidate items using (a) Sequential Pattern Mining Process by mining semantic rich frequent sequential patterns, (b) Collaborative Filtering item similarity matrix with semantic relationships between items, (c) Transition Probability Matrix in the Markov Model method and, (iii) generate semantic rich and sequential next item recommendations either by using semantic rich (a) Sequential Pattern Rules, (b) item based Collaborative Filtering or (c) Markov Model depending upon which method is used in step (ii)to generate candidate items.

1.4.2 Procedural Contributions

To address the above limitations and achieve the research goals, the procedural contributions according to the feature contributions are:
(i) Propose a model which first learns products' semantic representations in the form of multidimensional semantic feature vectors (embeddings) through various distribution models such as Prod2Vec (Grbovic et al., 2015), Glove(Pennington et al., 2014), Doc2vec(Mikolov et al., 2013), TF-IDF(Salton, 1988) and their hybrids by using customers' purchase histories and product meta data (title, description and brand) as explained in Chapter 5, Sect. 5.1 (Algorithm 2-Semantic Knowledge Extraction (SKE)). Next, cosine similarity between the obtained product feature vectors through these models is used to compute semantic similarity between products as products with similar vectors will be similar in semantics

and mapped closely in the vector space (Section 6.2.3). This semantic similarity is used to create an item-item similarity matrix for CF.

(ii) Next, to generate candidate items which are similar in semantics (in terms of their features or concept), we proposed to use any of these methods (Sequential Pattern Mining, Collaborative Filtering and Markov Model) by enriching these methods with the integration of the items' semantic knowledge in terms of (a) mining frequent semantic and sequential purchase patterns during the sequential pattern mining process by pruning patterns where items' similarity is less than the specified semantic similarity threshold in addition to the traditional pruning method of using minimum support (Chapter 6, Sect. 6.1, Algorithm 3-Semantic Knowledge Integration into SPM (SKI_SPM)), (b) enriching the item based Collaborative Filtering method with items' semantic and sequential relationships based on the proposed weighted score measure of products' score which considers semantic and sequential relationship along with confidence and lift measures between a pair of products and (c) enhancing the Transition Probability Matrix of the Markov Model based on proposed weighted product score measure which considers semantic and sequential relationships between a pair of products extracted from customers' purchase histories and products' meta data (Chapter 6, Sect. 6.2, Algorithm 4-Semantic and Sequential Knowledge Integration into Markov Model (SSKI_MM)).

(iii) In the recommendation phase, depending on which approach in step (ii) is used to generate candidate items, we recommend items that are similar in semantics and can be purchased in sequential order either by using (a) semantic rich frequent sequential rules (Chapter 7, Sect. 7.1, Algorithm 5-Semantic and Sequential Recommendation using Sequential Pattern Mining (SSR_SPM)) , (b) semantic and sequentially enhanced Collaborative Filtering matrix (Chapter 7, Sect. 7.1, Algorithm 6-Semantic Sequential Recommendation using Collaborative Filtering (SSR_CF)) or (c) semantic enhanced Markov Model's Transition Probability Matrix (Chapter 7, Sect. 7.2, Algorithm 7-Semantic Sequential Recommendation using Markov Model (SSR_MM)).

(iv) Furthermore, we also present a taxonomy of Sequential Recommendation Systems (SRecSys) with a focus on e-commerce product recommendation as an application, based on the techniques used to build these systems, along with the classification of surveyed systems according to eight important key features supported by the techniques, their limitations and future research directions.

1.5 Thesis Outline

The thesis is structured in the following way. Chapter 2 of this thesis presents the background relevant to achieve the research goals. It provides classification of RS based on (i) the input data sources used at the input phase and (ii) the techniques used for generating candidate items (users and items) by finding relationships between items and users at the candidate generation phase. Chapter 3 presents the problems identified in the existing systems, leading to the motivation behind this research. Chapter 4 states thesis problem statement along with the proposed system. Chapter 5, 6 and 7 each present details for an individual phase in the proposed system with Chapter 5 focused on learning semantic representation of e-commerce products, Chapter 6 presents details on extracting candidate items and users utilising item' semantic knowledge and Chapter 7 explains the process of generating semantic and sequential next item recommendations. Chapter 8 discusses the experiments performed, results obtained and their analysis. The thesis is concluded in Chapter 9 with possible future research directions to embark upon. Fig. 1.3 presents the flow and layout of Chapters 1 to 9 of this thesis.

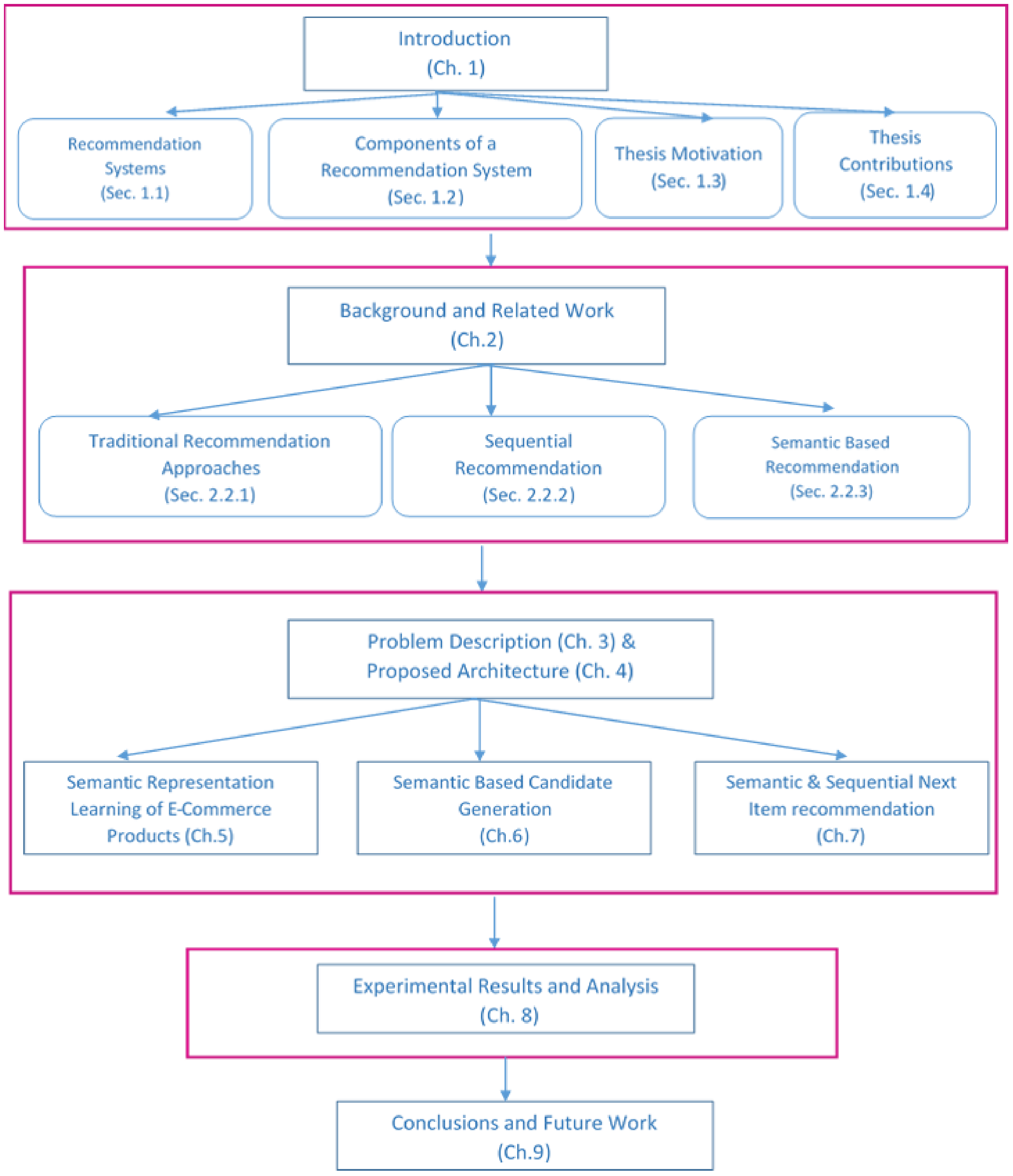

Introduction (Ch. 1)
Recommendation Systems (Sec. 1.1)
Components of a Recommendation System (Sec. 1.2)
Thesis Motivation (Sec. 1.3)
Thesis Contributions (Sec. 1.4)
Background and Related Work (Ch.2)
Traditional Recommendation Approaches (Sec. 2.2.1)
Sequential Recommendation (Sec. 2.2.2)
Semantic Based Recommendation (Sec. 2.2.3)
Problem Description (Ch. 3) & Proposed Architecture (Ch. 4)
Semantic Representation Learning of E-Commerce Products (Ch.5)
Semantic Based Candidate Generation (Ch.6)
Semantic & Sequential Next Item recommendation (Ch.7)
Experimental Results and Analysis (Ch. 8)
Conclusions and Future Work (Ch.9)

FIGURE 1.3: Thesis layout

Chapter 2

Related Work

This chapter discusses the background and the relevant work. Section 2.1 of this chapter provides classification of Recommendation Systems according to the usage of various input types, elaborating the pros and cons of each of these systems. Section 2.2 provides classification of RS according to various techniques used for candidate generation by finding relationships between items and users such as Similarity Based relationships (Content based and Neighborhood based), Sequential relationships (Traditional Approaches, Model Based and Neural Network Based) and Semantics Based relationships (Top down and Bottom up approaches). Each of these methods are discussed with running examples along with highlighting their pros and cons.

2.1 Classification of Recommendation Systems based on Input Data

Based on different types of information sources used to create user and item profile as discussed under traditional and auxiliary input sources (Section 1.2) we can classify recommendation systems into the following categories:

-Content Based- In these systems, recommendations are generated by using the descriptive attributes of items referred to as "content" along with user's implicit or explicit behavior (Lops et al., 2011). Consider an example of a customer Mary who highly rated the movie "Star Track" but we do not know the ratings of other users on this movie. However, we do know from the item description of "Star Track" that it has similar keywords from the science fiction genre as other movies like "Matrix" and "Terminator", so we can recommend these movies to Mary. These systems are well suited in scenarios where ratings of other users' are not available to find users with similar interests. The task is to create a vector based user profile on likes/dislikes or tags for the items she rated or consumed. Items' profile vector is also created for the same set of key words (descriptive features). Similarity is then computed between the user profile and items' profile to recommend items that a user may like. Algorithmic details and a walk through example for content based systems is provided in Section 2.2.1.

-Collaborative Filtering- These methods work on "collective" behavior to find similar users or items to predict ratings on an unseen item for the target user. In a user-item rating matrix R of size $u \times i$ where u represents users and i represents items and each entry in the matrix R_{ui} represents rating of user u on item i, (Bobadilla et al., 2013, Ekstrand et al., 2011, Schafer et al., 2007, Su and Khoshgoftaar, 2009). A row in the matrix represents a user profile vector representing user's explicit or implicit preferences towards items and each column represents an item profile vector representing interest of all users in the item. A user- based CF approach searches for neighbor users with similar preferences as the target user by computing row wise similarities between target user vector and each user vector, whereas item-based CF finds set of neighbor items by computing column wise similarity between target item vector and other item vectors. They model the user as a vector of item ratings (items with which the user interacted) and an item is modeled as a vector of user ratings (all users' who rated that item). Algorithmic details and walk through example is provided in Section 2.2.2.1.2.

-Knowledge Based- Knowledge based systems depend on domain knowledge for recommending items to users (Jannach et al., 2010). They are useful in domains where items are not purchased frequently such as automobiles, real estate and financial services. In such scenarios, user item ratings do not provide meaningful information for generating customized recommendations as it is very less likely to obtain sufficient ratings on an item which meets requirements of most users. For example, a car can have several features such as its make, engine, color and other interior features and a user's preference can depend on a specific combination of any of these available options. So, it is difficult to obtain sufficient ratings on the large number of item's available feature combinations. Hence, domain knowledge such as knowledge from experts in the automobile or finance sector can be helpful.

-Semantic Aware- Item profiles created from key words extracted from item descriptions or meta data associated with the item are insufficient to infer user's interests. Content extracted from textual features (item descriptions) can be misleading while learning a user profile due to ambiguity in natural language. For example, "hot dog" when used together refers to steamed sausage sandwich in the context of eating whereas the individual words refer to completely different meanings. Semantic Aware recommenders (de Gemmis et al., 2015) infer the semantic (meaning) of the words by exploring semantic relationships between items through the use of concepts extracted either from knowledge sources (ontologies) or distributional hypothesis (context in which the words are used). For example, in case of e-commerce products items co-purchased or co-viewed together. Section 2.2.3.1 and 2.2.3.2 are dedicated to provide algorithmic details on top down and bottom up approaches for learning semantic representation of user and item profiles.

-Context Aware-Unlike the traditional recommendation systems which collects users' preferences as ratings, context aware systems also include the ***"contextual information"*** to understand users' preferences. Context represents a set of factors or situations under which a user interacted with an item. For example, time, location, surroundings, purpose of purchase, device and occasion while interacting with an item (Adomavicius and Tuzhilin, 2011). Each context factor can be characterized by a structure such as the time factor can be described as seconds, minutes, hours, days, months and years. Such contextual information can be gathered from users' implicit or explicit feedback. For example, a user's rating for a hotel during her summer vacation stay.

-Demographic Based –These systems utilize user's demographic information to recommend items. Common demographic characteristics are location, gender, age, education, work experience and family history. For example, many websites generating recommendations according to user's demographic profile such as recommendations for restaurants near you by considering user's location, recommending online courses by taking user's current education and areas of interest.

-Time Aware- These systems integrate temporal knowledge (timestamps) into the recommendation process. For example, the time when the item was purchased, time when an item was rated. Incorporating temporal knowledge is important to consider the freshness of users' preferences and how they have evolved over time (Wang et al., 2012).

-Social Recommender- Users tend to trust more on the recommendations from their friends (social network) than recommendations by unknown people. These systems collect social information of users such as facebook friends, followers on twitter, gather preferences of their friends from social networks and then utilize their ratings on items to generate recommendations for the target user (Carrer-Neto et al., 2012, Morris et al., 2010). A common example could be recommending movie to watch based on the movies recently watched by target user's friends or recommending products by collecting information on products bought by target users' friends and are similar to her interest.

-Hybrid- Hybrid systems are created by (a) combinations of various input sources and/or (b) composition of different mechanism such as combining two or more recommendation systems. This is done to overcome the limitations of one recommender system with the other. For example, hybridising collaborative filtering systems with content based systems to address the item cold start issue (Burke, 2007). Tables 2.1, 2.2 and 2.3 show summary of recommendation systems on the basis on information sources (input) along with sample examples and their pros and cons.

2.2 Classification of Recommendation Systems according to Candidate Generation (Recommendation Techniques)

This section discusses various methods available for generating candidate item(s) for recommendation by finding associations between items and users. How these associations are modeled, depends on the choice of a particular recommendation method. In this section, we will discuss about (i) Traditional Approaches (keyword based and neighborhood based), (ii) Sequential Approaches (traditional approaches, Factorization and Latent representation based and neural network based), and (iii) Semantic Based Approaches (Top down approaches and Bottom up Approaches). Table 2.1, 2.2 and 2.3 represents the classification

TABLE 2.1: Recommendation Systems categorization on the basis of information sources (input)

Category	Input Type	Example	Common/Existing Systems/Example	Pros and Cons
Content Based	. User ratings . Item features (Content)	. **Item content,** e.g., movie genres, product descriptions . **User Feedback** -likes or dislikes,ratings, item purchase frequ ency	**Books - Amazon** - Similar genres - Similar authors **Movies - Netflix** - Similar actors and directors **Clothing** - Stitch Fix - Similar style	. No need for other user's ratings . Do not suffer from item cold start as uses items' content for recommendation . Cannot recommend diverse/novel items . Difficult to combine features from multiple items together
Collaborative Filtering	. User ratings . Community ratings (other users' ratings)	. Implicit or explicit ratings for products,likes or dislikes for movies	**Amazon** -Users who bought this item also bought. . . (user based CF) -Other items similar to what you bought are. . . . . . (Item based CF)	. Item features are not required . Can produce diverse recommendations (based on other neighbor users' interest) . Suffer from cold start and sparcity issues (due to less number of ratings available on items)
Knowledge Based	. User ratings, constraints . Item attribute, domain knowledge	Domain knowledge about real estate in a particular city to address features like location, size,neighborhood	Common Applications: - Real Estate, Automobiles Travel Recommendations	. Domain Knowledge is required . Knowledge base need to be maintained and updated regularly for new rules

TABLE 2.2: Recommendation Systems categorization on the basis of information sources-Input (Continued from Table 2.1)

Category	Input Type	Example	Common/Existing Systems/Example	Pros and Cons
Context Aware	. User ratings . Community rating Contextual knowledge	Context information such as location, day of week (weekend or weekday, seasonal shopping behavior during holidays	. Purchase recommendations,promo offers during Christmas holidays will be different than recommendations otherwise	. Can improve more customized and tailored recommendations . Difficult to obtain data containing contextual information . Cannot address the cold start problem for the new user
Time Aware	. User ratings . Community rating . Temporal knowledge	Temporal aspect such as timestamps when a user-item interaction took place	Trip planning recommendations for summer season mostly include sights with beaches Customers' purchase records User ratings during hotel stay in vacations	. Temporal information is helpful to generate sequential recommendations . Cannot address the sparsity issue in the matrix.
Social Recommender	. User ratings . Community ratings .Knowledge from users' social network	Gathering data from social networks, for example target user's friends from facebook, tweets from followers on twitter	For example, user likes to watch action movies, by looking at her friends post about recently watched action movies, those movies can be recommended to the user	. Depends largely on data from user network . Cannot address the cold start and sparsity issues . Can provide diverse recommendations

TABLE 2.3: Recommendation Systems categorization on the basis of information sources-input (Continued from Table 2.2)

Category	Input Type	Example	Common/Existing Systems/Example	Pros and Cons
Semantic Aware	. User ratings . Community ratings User/Item semantic knowledge	Incorporating semantic (conceptual) meaning for items such as the semantic relationship between items by using external knowledge sources (ontologies) or distributional hypothesis (co-occurrence of items in a context)	For example, a semantic representation of products will predict that low fat milk and skimmed milk are semantically similar and more closer than low fat milk and chocolate.	. Can provide more detailed insights by extracting semantics (meanings) for items in terms of their usage or conceptual relationships. . Can provide more tailored recommendations by inferring semantics from user profiles and items' characteristics
Hybrid	Combination of various inputs or composition of different mechanism	Combining input from various sources such as ratings and user demographics Combining different recommenders such as CF systems with content based systems	For example, having information about user's demographics such as gender, age, marital status and location can provide more customized product recommendations	. Can provide diverse recommendations . Can overcome cold start issues . Difficult to maintain the right balance (combination) of multiple models, e.g., the choice of optimal parameters (weights) for different models

of various recommendation techniques as presented in this chapter.

2.2.1 Traditional Approaches for Recommendation

These techniques generate recommendations either by using the descriptive attributes of items referred to as “content” along with user’s implicit or explicit behavior or by working on “collective” behavior to find similar users or items to predict ratings on an unseen item for the target user. In this section we will discuss about (i) keyword based methods (TF-IDF) and (ii) neighborhood based methods (collaborative filtering). The subsequent sections will discuss each of these techniques in detail.

2.2.1.1 Term Frequency-Inverse Document Frequency (TF-IDF)

This approach uses keyword matching or Vector space Model (VSM) (Lee et al., 1997) for representing items (documents, products, videos, music). VSM refers to spatial representation of items in which each item is represented as an n-dimensional vector. A dimension in the vector represents a term from the vocabulary obtained from a given collection of items. For example, in case of a large collection of text documents, a vocabulary will comprise of all possible unique words or consists of words (terms) that are more related to the content of the document. So, n will represent the number of unique terms for the whole document collection.

More specifically, each item (for example, a document) is represented as an n-dimensional vector of term weights, where each weight shows an association between the document and the term. If $d = \{d_1, d_2, \ldots\ldots, d_N\}$ denote a set of documents and $T = \{t_1, t_2, \ldots\ldots, t_N\}$ be the vocabulary, that is the set of words in the documents where T is obtained by applying some standard natural language processing operations. Some of these operations include (a) tokenization (the process of segmenting text into words, clauses or sentences such as separating words and removing punctuations), (b) stop words removal (removal of commonly used words unlikely to be useful for learning such as a, the, of), and (c)stemming which involves reducing related words to a common stem such as reducing the words loved, loving and lovely to the word love obtained from customer’s review about a product expressing her likeliness towards the product. Each document d_j is then represented as a vector in a

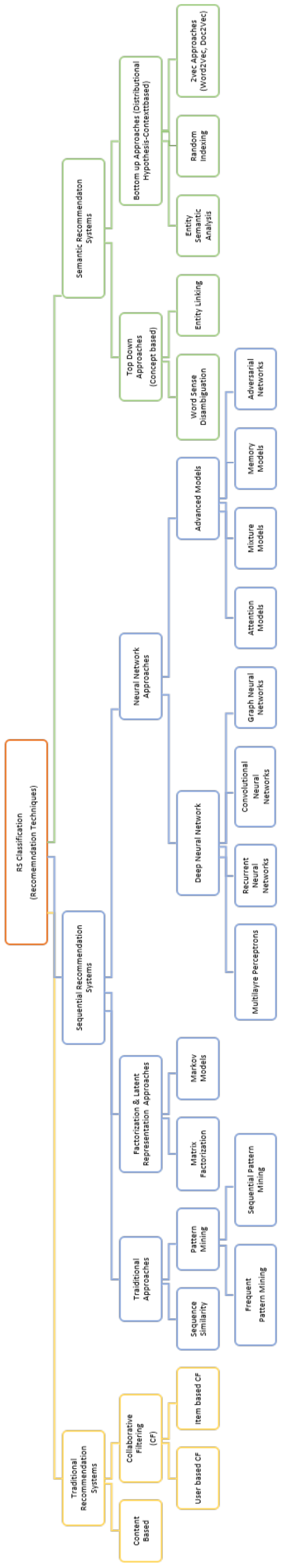

RS Classification
(Recomemndation Techniques)
Traditional Recommendation Systems
Content Based
Collaborative Filtering (CF)
User based CF
Item based CF
Sequential Recommendation Systems
Traiditional Approaches
Sequence Similarity
Pattern Mining
Frequent Pattern Mining
Sequential Pattern Mining
Factorization & Latent Representation Approaches
Matrix Factorization
Markov Models
Neural Network Approaches
Deep Neural Network
Multilayre Perceptrons
Recurrent Neural Networks
Convolutional Neural Networks
Graph Neural Networks
Advanced Models
Attention Models
Mixture Models
Memory Models
Adversarial Networks
Semantic Recommendaton Systems
Top Down Approaches (Concept based)
Word Sense Disambiguation
Entity Linking
Bottom up Approaches (Distributional Hypothesis-Contexttbased)
Entity Semantic Analysis
Random Indexing
2vec Approaches (Word2Vec, Doc2Vec)

Figure 2.1: Classification of Recommendation Systems according to candidate generation (Recommendation Techniques)

n-dimensional vector space, such that $d_j = < w_{1j}, w_{2j}, \ldots, w_{nj} >$ where w_{kj} is the weight for term t_k in document d_j.

After the creation of vocabulary also known as the features, next task is to represent each item (document in this case) in the VSM by weighting the terms and then measuring the similarities between feature vectors of items in order to find items similar to the target. TF-IDF weighting (Term Frequency-Inverse Document Frequency) is a term weighting scheme (Salton, 1988) which computes weight of a term as product of tf weight and idf weight, where tf is number of times the term occurs in the document and idf depends on rarity of a term in a collection. The advantage of this approach is tf-idf increases with the number of occurrences within a document, and with the rarity of the term in the collection. In other words, terms that occur frequently in one document (TF=term frequency), but rarely in the rest of the collection (IDF=inverse-document-frequency), are more likely to be relevant to the topic of the document. The TF-IDF of a term is computed using Eq. 2.1.

$$TF - IDF(t_k, d_j) = TF(t_k, d_j). \log N/n_k \tag{2.1}$$

where N represents the number of documents in the collection, and n_k denotes the number of documents in the collection in which the term t_k occurs at least once. Frequency of term t_k in document d_j is represented as $f_{tk,dj}$ and the maximum is computed over the frequencies $f_{z,j}$ of all terms t_z that occur in document d_j using Eq. 2.2.

$$TF\left(t_k, d_j\right) = \frac{f_{t_k,d_j}}{\max_z f_{z,j}} \tag{2.2}$$

To adjust the weights for falling in the interval [0, 1] interval and for the documents to be represented by equal length vectors, weights are usually normalized. Next, to determine the closeness between documents, similarity between their feature vectors is computed using cosine similarity by using Eq. 2.3.

$$\operatorname{sim}\left(d_i, d_j\right) = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}} \tag{2.3}$$

Table 2.4: Categorization of books into different genres

Book Title\Genre	Literature	Comedy	Autobiography	Science Fiction
King Lear	1	0	0	0
Macbeth	1	0	0	0
The invisible man	0	0	0	1
Long Walk to Freedom	0	0	1	0
Pride and Prejudice	1	0	0	0

In a similar fashion, in content-based recommender systems where both user profiles and items are represented as weighted term vectors using VSM, a user's interest in a particular item can be predicted by calculating the cosine similarity between them. Walk through examples (Example 2.1 and Example 2.2) explains the similarity computation by vector dot product and TF-IDF methods.

Example 2.1. TF-IDF (With Binary Values):

Consider an example, a user purchased the books "King Lear", "Macbeth" and "The invisible man" and provided ratings on a 0 to 5 scale as 3, 4, and 2 respectively. Considering the genre of books, their corresponding feature vectors can be represented in the binary form where a '1' shows that the book belongs to a particular genre and '0' means it does not. Assume, we have four genres which can be represented as four dimensions and shown in Table 2.4.

User's profile vector on these genres according to his ratings on the books will be (4,0,0,2) where '4' represents his interest in books from literature, '0' shows that he is not interested in reading books from comedy and romance genre and '2' shows his level of interest in science fiction books. Now, if we have to decide which new book to recommend to user from the books "Long Walk to Freedom" which is an autobiography and "Pride and Prejudice" which is a novel, we will compute the dot product (vector multiplication) between the user profile vector and the item (book) profile vector as follows:
dot product (user, Long Walk to Freedom)=(4,0,0,2) . (0,0,1,0) = (4*0 + 0*0 + 0*1 + 2*0)= 0
dot product (user, Pride and Prejudice) = (4,0,0,2) . (1,0,0,0) = (4*1 + 0*0 + 0*0 + 2*0) = 4.

Table 2.5: Frequency count of common terms occurring in the articles

	Terms						User Ratings	
Articles	Big Data	Python	R	ML	C++	Total	U1	U2
Article 1	0	24	0	2	2	28	1	-1
Article 2	24	0	2	1	0	27	-1	1
Article 3	0	35	8	10	19	72	1	0
Article 4	4	0	5	0	1	10	1	0
Article 5	0	48	4	3	4	59	0	1
Article 6	0	0	8	0	5	13	0	0

The result of the dot product is “0” for the book “Long Walk to the Freedom” which shows that the user is not interested in reading autobiographies and the dot product result for “Pride and Prejudice” is 4 which shows that user will likely enjoy this literary novel by Jane Auston, so this will be recommended to the user. Another possibility can be to represent the contents of each book by extracting key words from their description and then computing similarities based on those key-word based profiles.

Problem: Given a collection of documents and a set of features (terms), find the association (weight) between the document and the features.

Example 2.2. TF-IDF (With Frequency Count):

Consider another example, representing a collection of articles related to programming languages, representing the frequency count (occurrence) of terms across each document. The steps to create the item and user profiles using TF-IDF and then computing the similarities between user and item profiles for article recommendation are given below:

Table 2.5. shows frequency count of five terms occurring in the articles related to programming languages. For example, the term Python appears 24 times in Article 1 and 48 times in Article 5. The table also shows the ratings of user u1 and u2 on these articles, such as, a rating of ‘1’ shows that the user liked the article and ‘-1’ means she disliked it.
Step 1: Term Frequency Computation (TF)

Term Frequency (TF) will be computed by using Eq 2. For example term frequency for Python in Article 1 is computed as : TF (Python, Article 1) = 24/28 =0.86. Table 2.6.

Table 2.6: Term Frequency computation

	Terms				
Articles	Big Data	Python	R	ML	C++
Article 1	0	0.86	0.00	0.07	0.07
Article 2	0.89	0	0.07	0.04	0
Article 3	0	0.49	0.11	0.14	0.26
Article 4	0.40	0	0.50	0	0.10
Article 5	0	0.81	0.07	0.05	0.07
Article 6	0	0	0.62	0	0.38

Table 2.7: Inverse Document Frequency (IDF) computation

Terms	Big Data	Python	R	ML	C++
IDF	0.48	0.30	0.08	0.18	0.08

shows the term frequencies computed for all the terms.

Step 2: Inverse Document Frequency Computation (IDF)

IDF for all the terms is computed using the formula (log N/nk) where N is the total number of articles and nk represents the number of documents in which the term appears at least once. Table 2.7.shows the IDF of all terms.

Step 3: TF- IDF Computation

The TF-IDF is computed using Eq. 2.1., that is, taking the product of Term Frequency of each term in the article with Inverse Document Frequency of the term in that article. For example, the TF-IDF of the term Python will be:

TF-IDF (Python, Article 1) = TF (Python) * IDF (Python, Article 1) = 0.86 * 0.30 = 0.26

After the computation of TF-IDF (Table 2.8), we can represent each article as an n-dimensional feature vector, where each dimension represents a feature (term). For our

Table 2.8: TF-IDF of terms in articles

	Terms				
Articles	Big Data	Python	R	ML	C++
Article 1	0	0.26	0	0.01	0.01
Article 2	0.42	0	0.01	0.01	0.00
Article 3	0	0.15	0.01	0.02	0.02
Article 4	0.19	0	0.04	0	0.01
Article 5	0	0.24	0.01	0.01	0.01
Article 6	0	0	0.05	0	0.03

example, we have five features which are Big Data, Python, R , ML and C++. Article 1 can be represented as term weighted vector as:
Article 1 = [0, 0.26, 0, 0.01, 0.01]
which shows that Article 1 talks more about the Python programming language and is weighted more towards the Python language.

Step 4: Similarity Computation between Articles

After we have each article represented in the vector form (each row in Table 2.8. represents vector for the corresponding article), we can compute the closeness between the articles by computing the cosine similarities using Eq. 2.3 between their vectors. For example, some cosine similarities are Cos (A1, A2) = 0.00013, Cos (A1, A3)= 0.0221, which shows Article 3 is more closely related to Article 1. In the similar way, we can compute the similarity between the user profile and the article's profile to predict which article to recommend to the user.

2.2.1.2 Neighborhood Based

These methods work on "collective" behavior to find similar users or items to predict ratings on an unseen item for the target user. Collaborative Filtering (CF) is the most common type of recommendation systems which takes users' interest in an item as input and store it in a matrix known as user-item rating matrix. The system outputs top recommendations for the target user by finding users with the similar interest (Ekstrand et al., 2011). In CF recommendation systems, the users only rate a small number of items, therefore, the rating matrix is very sparse and it is necessary to predict the missing ratings to seek more accurate recommendation results for users. In a user-item rating matrix R of size $u \times i$ where u represents users and i represents items and each entry in the matrix R_{ui} represents rating of user u on item i. Common Collaborative Filtering systems are (a) user-to-user and (b) item-to-item. In the next subsection we will discuss each of these approaches along with a walk through example.

-User based CF- Each row in the user-item matrix represents a user vector representing user's explicit or implicit preferences towards items and each column represents an item vector representing interest of all the users in the item. A user- based CF approach

Table 2.9: User-Item rating matrix

	Item 1	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**
User A	7	6	7	4	5	4
User B	6	7	?	4	3	4
User C (Alice)	?	3	3	1	1	?
User D	1	2	2	3	3	4
User E	1	?	1	2	3	3

searches for neighbor users with similar preferences as the target user by computing row wise similarities between target user vector and each user vector, whereas item-based CF finds set of neighbor items by computing column wise similarity between target item vector and other item vectors. Consider the below examples (Aggarwal, 2016) to understand user based (Example 2.3) and item based CF (Example 2.4). Table 2.9. shows the user-item rating matrix containing ratings of five users on five items. User C (Alice) is the target user and we are interested to predict rating for her on item 1 which she has not consumed/rated before.

Problem: Given a user-item rating matrix R, predict the unknown ratings on items for the user .

Example 2.3. User-to-User CF

Table 2.10: User-Item rating matrix with mean ratings and similarity scores for user based CF

	Item 1	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**	**Mean Ratings**	**Similarity Score**
User A	7	6	7	4	5	4	33/6 =5.5	0.95
User B	6	7	?	4	3	4	24/5 =4.8	0.98
User C (Alice)	?	3	3	1	1	?	8/4= 2	
User D	1	2	2	3	3	4	15/6= 2.5	0.78
User E	1	?	1	2	3	3	10/5 =2	0.64

1. Compute the mean rating for each user u_j using all of their rated items. Mean rating of User A $= ((7+6+7+4+5+4))/6 = 33/6 = 5.5$. Similarly, User B=4.8, User C=2, User D= 2.5 and User E=2. Mean ratings are shown in Table 9.

2. Calculate the similarity between a target user v and all other users u_j. Similarity can

Table 2.11: User-Item rating matrix with mean ratings and similarity scores for item based CF

Users\Items	**Item 1**	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**
User A	7	6	7	4	5	4
User B	6	7	?	4	3	4
User C (Alice)	?	3	3	1	1	?
User D	1	2	2	3	3	4
User E	1	?	1	2	3	3
Mean Item Ratings	3.75	4.5	3.25	2.8	3	3.75
Similarity score		0.99	0.99	0.91	0.83	0.839

be computed with $CosineSimilarity(v, u_j)$ or other similarity functions.

$$\mathrm{SIM(u,v)} = \frac{r_{u1} * r_{v1} + r_{u2} * r_{v2} + \ldots + r_{un} * r_{vn}}{\sqrt{r_{u1}^2 + r_{u2}^2 + \cdots + r_{un}^2} * \sqrt{r_{v1}^2 + r_{v2}^2 + \cdots + r_{vn}}} \tag{2.4}$$

Where, r_{u1} represents rating given by user u on item 1 and r_{v1} represents rating given by user v on item 1. In our case, SIM (Alice, User A) = $(6 * 3 + 7 * 3 + 4 * 1 + 5 * 1)/(\sqrt{(6^2 + 7^2 + 4^2 + 5^2)} * \sqrt{(3^2 + 3 + 1^2 + 1^2)}) = 0.956$. Similarly, SIM (Alice, User B) =0.981, SIM (Alice, User D) =0.789 and SIM (Alice, User E) =0.645.

3. Find similar users of target user v as their Top-N users. For example, if we want to select Top-2 users to Alice, then in our case, User A and User B have the highest similarity with Alice compared to other users.

4. Predict rating for target user v for item i using only ratings of v's Top-N peer group. Predict rating of Alice on Item 1 using rating of Top-2 users (User A and User B) on Item 1.

$r_{userC,item1} =$

$$\frac{r_{userA,item1} * SIM(Alice, UserA) + r_{UserB,Item1} * SIM(Alice, UserB)}{SIM(Alice, UserA) + SIM(Alice, UserB)} \tag{2.5}$$

-Item Based CF-

Problem: Given a user-item rating matrix R, predict the unknown ratings on items for the user.

Example 2.4. Item-to-Item CF

1. Compute the mean rating for each item i_j.

   Mean rating of Item 1= $((7+6+1+1))/4 = 3.75$. Similarly, for other items as shown in Table 2.11.

2. Calculate the similarity between a target item i for target user Alice and all other items j_n. Similarity can be computed with $CosineSimilarity(i, j_n)$ or other similarity functions.

   $Sim(i, j) =$
$$\frac{r_{i1} * r_{j1} + r_{i2} * r_{j2} + \ldots + r_{in} * r_{un}}{\sqrt{r_{i1}^2 + r_{i2}^2 + \cdots + r_{in}^2} * \sqrt{r_{j1}^2 + r_{j2}^2 + \cdots + r_{jn}^2}} \tag{2.6}$$

   Where, r_{i1} by user A and r_{j1} represents rating on item 2 given by user A. In our case, $SIM(item1, item2) = (7*6+6*7+1*2)/(\sqrt{(7^2+6^2+1^2)}*\sqrt{(6^2+7^2+2^2)}) = 0.99$. Other scores are shown in Table 2.11.

3. Find similar items of target item 1 as its Top-N items. For example, if we want to select Top-2 similar items to item 1, then in our case, item 2 and item 3 have the highest similarity with item 1.

4. Predict rating for target item 1 for user Alice using only ratings of item j's Top-N peer group. Predict rating of item 1 for Alice using ratings of Top-2 items (Item 2 and Item 3).

   $r_{userC,item1} =$

$$\frac{r_{Alice,Item2} * SIM(Item1, item2) + r_{Alice,Item3} * SIM(item1, item3)}{SIM(item1, item2) + SIM(item1, item3)} \tag{2.7}$$

   $= (3 * 0.99 + 3 * 0.99)/(0.99 + 0.99) = 3.04$

2.2.2 Sequential Recommendation Techniques

This section will review related approaches for learning sequential relationships between items for next item recommendation. The sequential relationships assist in modeling user preferences by reflecting the sequential dependency of an event (e.g., click or purchase) on its preceding event. The approaches are categorized as (i) Traditional Approaches (Sequence Similarity, Frequent Pattern Mining and Sequential Pattern Mining), (ii) Factorization and

Latent Representation Approaches (Matrix Factorization and Markov Models) and (iii) Neural Network Based Approaches (Deep Neural Network, Advanced Models). The subsequent sections will thoroughly discuss each of these techniques along with providing details on their theoretical aspects, potential for solving problems and their limitations.

2.2.2.1 Traditional Approaches for Sequential Recommendation Systems

To learn sequential relationships between items, these approaches work either by computing similarities between sequences of items (e.g., click or purchase sequences) according to a similarity measure (similarity based) or by mining sequential patterns that occur frequently in the data. Section 2.2.2.1.1 discuss about sequence similarity based approaches while Sections 2.2.2.1.2 and 2.2.2.1.3 will review pattern mining based methods including frequent pattern mining and frequent sequential pattern mining methods respectively. We also present a classification of traditional approaches for Sequential Recommendation Systems on the basis of eight key features as discussed in Sect. 2.2.2.4

2.2.2.1.1 Sequence Similarity Based Methods The objective of these approaches is to find relationships between items or sequences by computing similarity based on a similarity measure and then utilizing sequences similar to the target user sequence for next item recommendation. These relationships reflect the sequential dependency of an event (e.g., click or purchase) on its preceding event. The granularity of sequential dependency can be at item level (single item) or sequence level (comprising of multiple item sets, where each item set can be a single item or collection of items).

Problem: Find relationships between item sequences by computing similarity using a similarity measure and then utilizing sequences similar to the target user sequence for next item recommendation.

Problem: Find relationships between item sequences by computing similarity using a similarity measure and then utilizing sequences similar to the target user sequence for next item recommendation. For example, consider a purchase sequence of a customer as $< (a, b), (b, c, a, d), (a), (c, d, d) >$ which shows items purchased by customer on different timestamps. The sequence consists of four item sets which are $(a, b), (b, c, a, d), (a)$ and (c, d, d). The item set (a, b) consists of two items a and b which were purchased together and the item set a just represents single item which was purchased individually in another transaction. Sequence similarity can be computed by finding the longest common subsequence (LCS) rate between the two sequences and then utilizing the obtained rate to recommend items from the sequence with the highest rate to the target sequence. Consider purchase sequences of two customers as $X = <(a, b), (b, c, a, d), (a), <(c, d, d)>$ and $Y = <(a, a, b), (b, c, c), (a, b)>$, the LCS rate (Bhatta et al., 2019, Hunt and MacIlroy, 1976) between sequence X and Y can be computed as:

$$\begin{aligned} LCSR(X,Y) &= \frac{\text{LCS}(X,Y)}{\max(X,Y)} \\ &= \frac{(a,b),(b,c),(a)}{(a,b),(b,c,a,d),(a),(c,d,d)>} = \frac{5}{10} = 0.5 \end{aligned} \tag{2.8}$$

where LCS (Yap et al., 2012) can be computed as:

$$LCS(X,Y) = \begin{cases} \phi \quad \text{if } i = 0 \text{ or } j = 0 \\ \text{LCS}\left(X_{i-1}, Y_{j-1}\right) \cap X_i \quad \text{if } x_i = y_i \\ \text{longest } \left(LCS\left(X_i, Y_{j-1}\right), \text{LCS}\left(X_{i-1}, Y_j\right) \quad \text{if } x_{i \neq} y_i \end{cases} \tag{2.9}$$

Another method of computing similarity between sequences is by representing each sequence as a vector of events (e.g., item's purchased). Considering the same purchase sequences X and Y, their corresponding vectors in terms of the frequency of items purchased will be $\overrightarrow{X}$= [a b c d] = [3 2 1 3] and $\overrightarrow{Y}$ = [a b c d] = [3 3 2 0] . The similarity between $\overrightarrow{X}, \overrightarrow{Y}$ is calculated by using cosine similarity as:

$$\begin{aligned} \text{sim}(X,Y) = \cos(\vec{X}, \vec{Y}) &= \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}} \\ &= \frac{(3*3)+(2*3)+(1*2)+(3*0)}{\sqrt{3^2+2^2+1^2+3^2} * \sqrt{3^2+3^2+2^2+0^2}} = 0.87 \end{aligned} \tag{2.10}$$

A sequential recommendation system, ChenRec15 (Su and Chen, 2015) finds the similarity between two users based on their clickstream sequences. They measure indicators like category visiting path, category browsing frequency and category access time, then cluster users into different clusters, by selecting top-N similar users, and then provide better top-N recommendations. However, they only focus on the category level visits, and their technique for mining the whole dataset is not very efficient. To improve the effectiveness of recommendation systems by incorporating the sequence information about user's item ratings, a system is proposed (Cheng et al., 2016). To depict users' dynamic interest evolution patterns, (Cheng et al., 2016) uses the concept of "interest sequences" (IS) which takes the sequences of users' behaviors based on the time of their interactions (ratings). Assume two users u and v. User u rated items C, E and D at timestamps t_1, t_2 and t_3 with ratings 4.0, 3.5 and 2.0 respectively. User v rated the same items but in different order as items E, D, C at timestamp t_1, t_2 and t_3 with ratings 2.0, 4.5, 2.5. So, IS of user u (i_{su}) according to timestamp t_1, t_2 and $t_3 = (C, 4.0), (E, 3.5), (D, 2.0)$. Similarly, IS of user v at these time stamps will be $i_{sv} = (E, 2.0), (D, 4.5), (C, 2.5)$. Next, an Interest Sequence Match (ISM) between users is calculated if (a) the rated items between them are same and (b) the deviation between the ratings given by the user u and user v for the same item should be less than a specified threshold θ. Using these interest sequences, they computed the length of Longest Common Sub-IS (LCSIS) (which refers to an Interest Sequence Match (ISM) between two interest sequences if and only if there is no other longer ISM detected between them) and the count of All Common Sub-IS (ACSIS) which is used to count all the ISM between two interest sequences, including empty ISM. Considering the length of LCSIS and the count of ACSIS, users' similarities are computed based on IS, which are then used to find the Top-K nearest neighbors for the target user.

HPCRec18 (Xiao and Ezeife, 2018) used normalized purchased frequency matrix to improve the quality of ratings by mining the consequential bond (similarity between click and purchase sequences) in each session to predict the ratings for next possible purchase. The proposed system (Xiao and Ezeife, 2018) enriched the quantity (finding the value for unknown ratings) and quality (finding precise value for already rated items) of user-item rating matrix. However, the main limitation was that customers' historical sequential behavior patterns were not integrated into the item rating matrix and during the mining process of consequential bond. This limitation was addressed by HSPRec19 (Bhatta et al., 2019) which incorporated sequential information while mining the consequential bond and

Table 2.12: Implicit user-item rating matrix

User\Items	Apple	Orange	Banana	Bread	Butter	Milk
User1	0	0	0	1	1	1
User2	0	1	0	0	1	1
User3	0	0	0	1	1	0
User4	1	1	1	1	1	1
User5	1	0	1	0	0	0
User6	1	1	1	0	0	0
User7	1	1	0	0	1	0

enriching the user-item matrix.

2.2.2.1.2 Frequent Pattern Mining The objective of frequent pattern mining is to find relationships (patterns) between items that occur frequently in the transaction database and then utilizing the mined patterns to guide the subsequent next item recommendations. Association rules (Agrawal and Srikant, 1995) were first introduced to discover relationship between items in supermarket data. Consider a transaction database of customer purchases, $T = \{T_1, T_2, \ldots, T_m\}$, containing m transactions, which are defined on items I. Each transaction T_i is a subset of the items in I. To find these relationships between items, two measures, support and confidence are used. The support of an item is the measure of the frequency (occurrence) of the item in the transaction database. A minimum support threshold s is used to determine if the item is frequent or not. An item is said to be a frequent item if its support is greater than or equal to the minimum support s. These frequent items or frequent patterns can be useful in providing valuable insights about customers' buying behavior. For example, consider the data from customers' purchase in Table 2.12. The rows correspond to customers and columns correspond to items. The 1s refer to the situation in which a particular customer has bought an item.

These are partitioned into two sets of closely related items. One of these sets is $\{Apple, Orange, Banana\}$ and the other is $\{Bread, Butter, Milk\}$. These are the only itemsets with at least 3 items, which also have a support of at least 0.2. Therefore, both of these itemsets are frequent itemsets or frequent patterns. Finding such patterns with high support is useful to the merchant, because he can use them to make recommendations and other targeted marketing decisions. For example, it can be concluded that User2 is likely to eventually buy $Bread$, because he has already bought $\{Butter, Milk\}$. Similarly, User5 is likely to buy *oranges* because he has also bought $\{Apple, Banana\}$. Such inferences are very useful from the point of view of a Sequential Recommendation System. Another insight can be obtained in terms of the directions of these correlations by using the concept of association rules and confidence. An association rule is denoted in the form $X \rightarrow Y$, which shows the correlation between the set of items X and Y. For example, a rule such as $\{Butter, Milk\} \rightarrow \{Bread\}$ would be very useful to recommend *Bread* to User2, because it is already known that he has bought *Milk* and *Butter*. The strength of such a rule is measured by its confidence. The confidence of the rule $X \rightarrow Y$ is the "conditional probability that a transaction in T contains Y, given that it also contains X". Higher values of the confidence are always indicative of greater strength of the rule. An association rule is defined on the basis of a minimum support s and minimum confidence c. One of the earliest works in frequent pattern mining and association rule mining is Apriori algorithm (Agrawal and Srikant, 1995). Given a transaction database, a minimum support threshold s, the algorithm mines the transaction database to find frequent itemsets for association rules. The algorithm is based on two important steps which are (a) join and (b) prune. Below are the steps for generating candidate items using Apriori algorithm (Agrawal and Srikant, 1995).

1. Generate all singleton (one-item) candidate items (C_k) from the transaction database.
2. Prune any items that do not meet the minimum support threshold s. This gives the frequent L_k items.
3. Generate 2- item set candidate items C_{k+1} by using the apriori property of L_k join L_k.
4. Repeat step 2 to generate L_{k+1} frequent item sets.
5. The process is repeated until there are no further candidate itemsets. The apriori algorithm works on the apriori property, that is, any $(k-1)$-itemset that is not frequent cannot be a subset of a frequent k-itemset.
6. The set of frequent itemsets L will be the union of all $L_k, L_{k+1}, \ldots L_n$. The above steps can be explained with the help of a running example (Example 2.5).

TABLE 2.13: Customer's transaction database

TID	Items
001	{Bread, Butter, Jam}
002	{Bread, Eggs, Butter}
003	{Bread, Eggs}
004	{Milk, Tea}
005	{Bread, Butter, Eggs, Cheese}

Example 2.5.Association Rule Mining Using Apriori Algorithm

Consider Table 2.13. which shows four transactions by customers. Each transaction is represented by a transaction id (TID) and the set of items purchased in that transaction. For example, transaction 001 shows that $Bread, Butter$ and Jam were purchased. Given these transactions and a minimum support $s = 2$, which means an item should appear at least twice to be considered as frequent, the algorithm works by finding all singleton candidate items. The one-items candidate items C_k with their support count are $\{Bread : 4, Butter : 2, Eggs : 2, Tea : 1, Milk : 1, Jam : 1, Cheese : 1\}$. After the pruning step, that is, elements not meeting the minimum support which are $\{Tea, Milk, JamandCheese\}$, we have the frequent L_k items as $\{Bread : 3, Butter : 3, Eggs : 3\}$. Next to create 2-items set candidate items C_{k+1}, we will join each L_k with itself which gives the items $\{Bread, Butter : 3\}, \{Bread, Eggs : 3\}, \{Butter, Eggs : 2\}$. Repeating the steps, we will get the final set of 3-item frequent items as $\{Bread, Butter, Eggs : 2\}$. Since, we do not have any more candidate items to join, the algorithm will terminate. The set of frequent item sets is $L = \{\{Bread\}, \{Butter\}, \{Eggs\}, \{Bread, Butter\},$
$\{Bread, Eggs\}, \{Butter, Eggs\},$
$\{Bread, Butter, Eggs\}\}$.

Generating rules from frequent items:

The rules are generated by following these steps:

1. For each frequent itemset l, generate all nonempty subsets of l.

2. For every nonempty subset s of l, output the rule $s \to (l-s)$ if support_count (l) / support_count(s) >= min_conf, where min_conf is the minimum confidence threshold. From our example, if we take the itemset $I = \{Bread, Butter, Eggs\}$, its all nonempty subsets are $\{Bread, Butter\}$, $\{Bread, Eggs\}$, $\{ButterEggs\}$,$\{Bread\}$,$\{Butter\}$,

$\{Eggs\}$. Given a minimum confidence of 0.5, some of the rules can be:

(a) $\{Bread, Butter\} \rightarrow Eggs$ and

(b) $\{Bread, Eggs\} \rightarrow Butter$ with a confidence of 0.6.

Kim11Rec (Kim and Yum, 2011), a recommendation system used association rule mining and calculated confidence between products at different levels such as clicks, basket placement and purchases. The main idea is to find the most relevant clicked item. This is repeated for all the other stages i.e., basket places and purchases. A final score is calculated by assigning weights to the scores of the above mentioned three stages. The disadvantage of association rule methods is that it does not include connections between different users who share special interests (Kim and Yum, 2011). These systems simply measure the interest by browsing paths from the clickstream data.

2.2.2.1.3 Frequent Sequential Pattern Mining These techniques mine patterns which occur frequently in a sequential database and then use the mined sequential patterns for generating subsequent recommendation. A common application is mining customers' purchase sequence database for generating frequent sequential purchase patterns which can predict next item recommendations. A sequence database consists of ordered elements or events (Bhatta et al., 2019, Mabroukeh and Ezeife, 2010). Formally stated,
Given (i) a set of sequential records (called sequences) representing a sequential database SDB= $\{s_1, s_2, s_3, \ldots, s_n\}$ with sequence identifiers 1,2,3,. . . .n; (ii) a minimum support threshold called min sup ξ and (iii) a set of k unique items or events $I = \{i_1, i_2, ..., i_k\}$. The goal of mining sequential patterns is to find the set of all frequent sub-sequences S in the given sequence database SDB of items I at the given min sup ξ, that are interesting for the user. Consider the example in Table 2.14. showing a sequence database containing sequences representing four transactions made by four customers at a retail store in a week. Items between curly brackets represent an item set purchased in one market visit. For example, the sequence with SID 3 indicates that a customer first purchased *bread* in one transaction, then *milk*, followed by the purchase of *egg* and *butter* in the same transaction and lastly cream was purchased in another transaction.

Problem: Given an input of a sequential database (SDB) and a minimum support, the goal of sequential pattern mining algorithms such as GSP (Agrawal and Srikant, 1995) is

Table 2.14: A sequence database (SDB)

SID	Sequence
1	{bread, milk},{salt},{egg, butter},{butter}
2	{bread, cheese},{salt},{milk},{bread, milk, cream}
3	{bread},{milk},{egg, butter},{cream}
4	{milk},{egg, butter}

to mine all frequent sequential patterns from a sequence database having the support count greater than or equal to the user defined threshold of minimum support.

A sequence s is said to be a frequent sequential pattern if and only if the support (number of occurrence of s) is greater than or equal to a given minimum support. One of the earliest sequential pattern mining algorithm GSP (Agrawal and Srikant, 1995) which works on the apriori principle while maintaining the order of items will generate some of the frequent sequences as $< \{bread\} >$, $< \{salt\}$, $< \{bread\},\{butter\} >$, $< \{bread, cream\} >$, $< \{bread, milk\} >$, if the given minimum support is 2. So, sequence $< \{bread, milk\} >$ is frequent as it occurs in two sequences (SID 1 and 2) in the sequence database. In this case, the sequence $< \{bread, milk\} >$ and $< \{milk, bread\} >$ will be considered different, as their order is different in the sequences.

Table 2.15: A sample sequential database (SDB)

SID	Sequences
1.	$< 21239, (21239, 20655, 21242), (21239, 21242), (21366), (21242, 22246) >$
2.	$< (21239, 21366), 21242, (20655, 21242), (21239, 21377) >$
3.	$< (21377, 22246), (21239, 20655), (21366, 22246), (21242, 20655) >$
4.	$< 21377, 22198, (21239, 22246), 21242, 20655, 21242 >$

Table 2.16: Projected Database of item $< 21239 >$

Pattern	Projected Database of prefix $< 21239 >$
$< 21239 >$	$< (21239, 20655, 21242), (21239, 21242), (21366), (21242, 22246) >$ $<\_, 21366), 21242, (20655, 21242), (21239, 21377) >$ $<\_, 22246), (21239, 20655), (21366, 22246), (21242, 20655) >$ $<\_, 22198, (21239, 22246), 21242, 20655, 21242 >$

Example 2.6. Mining Sequential Patterns using PrefixSpan (Jian Pei et al., 2001)

With purchase sequential database Table 2.15 and a min_ sup=1, PrefixSpan (Jian Pei et al., 2001) generates frequent sequential patterns as :

1. Scan the database to generate frequent sequences (s) of length one (L_1) by matching the support of each item with the minimum support. Prune any item whose support is less than the min_sup. Here L_1= {<21239>, <20655>,<21242>, <21366>, <21377>, <22246> }.

2. Divide the search space such that the complete set of sequential patterns can be partitioned into subsets. In our case, we will have six subsets (according to items in L1), for example, one with prefix <21239>, then with prefix <20655> and so on.

3. Create a projected database for each prefix. For example, Table 2.16 shows projected database of item <21239>. Next, to generate length-2 sequential patterns of s, scan the projected database of s (if it is not null) and find the set of frequent items x such that: (a) x can be assembled to the last element of s to form a sequential pattern or, (b) can be appended to s to form a sequential pattern. For each frequent item x, append it to s to form a sequential pattern s', and output s'. For example, some of the frequent length-2 sequences for s = <21239> are: <(21239,21239),<21239,20655>,<(21239,20655)>, <21239,21242>,<21239,21366>, <21239,22246>.

4. Continue the process of creating projected database for each s' to generate L_{k+1} sequences until there is an empty set. Repeat the process for the remaining items (s) in L_1 and output the set of all frequent sequences. In our running example, some of the frequent sequential patterns are:<21239,21242>, <21242,22246>, <20655,21242>, <(21366,22246)>, <21242,20655,21242>, <(21239,29655,21242)>.

Many Sequential Pattern Mining algorithms were introduced in the literature including GSP (Agrawal and Srikant, 1995), SPAM (Ayres et al., 2002), Prefix Span (Jian Pei et al., 2001) and PLWAP (Ezeife et al., 2005). An extensive surveys on various Sequential Pattern Mining algorithms for next item prediction is presented in (Fournier-Viger and Lin, Mabroukeh and Ezeife, 2010, Mooney and Roddick, 2013).

Other research works (Choi et al., 2012, Saini et al., 2017) also use Sequential Pattern Mining to explore the sequential relationship between purchase sequences for items recommendations. These models are hybrid as they combine Collaborative Filtering with Sequential Pattern Mining for providing recommendations. The idea is to find neighbor users not just on the basis of ratings on similar items but also considering the sequential relationship between purchase sequences which can lead towards more accurate recommendations. Some of the recent works (Bhatta et al., 2019, Choi et al., 2012, Saini et al., 2017, Salehi and Nakhai Kamalabadi, 2013, Yap et al., 2012) have utilized sequential pattern mining techniques in e-commerce recommendation systems for mining customers' purchase patterns and recommending next items for purchase. A hybrid online product recommendation system (HOPE) (Choi et al., 2012) was proposed, which integrates Collaborative Filtering (CF) based recommendations using implicit rating and sequential pattern mining based recommendations. The systems uses implicit rating information instead of explicit rating information by computing implicit rating information from the transaction dataset. The recommendation quality was improved by integrating Collaborative Filtering and sequential pattern analysis of customer purchases each of which considers the rating information of users on items and the associations among items (Choi et al., 2012).

In another study, Yap et al. (Yap et al., 2012) stated that most of the existing sequential pattern mining methods are not user-specific and proposed a personalized sequential pattern mining based approach using a novel Competence Score to overcome such disadvantages. More weight was assigned to sequences that had similar items/products as the target user (e.g., user's purchasing/viewing the same books as the target user). To mine sequences in the purchase patterns of customer transactions, a framework is proposed (Saini et al., 2017) which aims to find the time gap between products purchased to help improve the recommendation of next products in sequence. SPADE (Zaki, 2001) algorithm was used to mine sequences of all products which are 1) bought regularly e.g., every month and 2) bought one after another in a sequence, e.g., most users find mobile phones and mobile covers as common purchase sequence.

HSPRec19 (Bhatta et al., 2019) mined frequent sequential purchase patterns and augment the item rating matrix with sequential purchase patterns of customers for next item recommendation. It mined frequent sequential click and purchase behavior patterns using the consequential bond between click and purchase sequences and then used this quantitatively and qualitatively rich matrix for Collaborative Filtering to provide better recommendations. The results (Bhatta et al., 2019) have showed significant improvement over the systems without using sequential pattern mining methods such as (Kim and Yum, 2011, Kim et al., 2005, Su and Chen, 2015, Xiao and Ezeife, 2018). Therefore, mining sequential patterns from customers' historical transactions can be very beneficial for retailers and can increase revenue and customer satisfaction by recommending tailored products to customers.

In summary, rule based approaches suffer from some limitations in terms of generating patterns including deciding an optimal value for the minimum support threshold ξ, setting it to a value which is too low will result in unnecessary patterns and setting it too high may lead to very few patterns which may not be very useful. Furthermore, the patterns are not personalised, that is they do not take into account an individual's preferences while they are generated as they are based on the historical data of transactions from all customers. This also limits the capability of these systems as they can only recommend items which exist in the historical data (for example, items purchased by the customers) and meet the minimum support threshold ξ thereby not exploring thousands of products available in the catalog for recommendation. Additionally, the generated patterns and subsequent rules lack semantics, that is, the relationship between items is based on their co-occurrence (frequency count) only, hence these approaches cannot infer the underlying relationships between items, for example, based on our common knowledge, we can understand that "low fat milk" and "skimmed milk" have same properties. So, if a user is conscious about her calorie intake and consumes low fat milk, then by inferring these semantic relationships between items, we can recommend her other low calorie products. Existing work utilizing sequential pattern mining approaches are unable to address these requirements.

2.2.2.2 Factorization and Latent Representation Based Approaches

These techniques create a model from the training data (observed user ratings) and then predict the ratings for users on unseen items based on the observations learned from the trained model. Model-based methods apply machine learning techniques during the model training process. Model-based methods are assumed to capture underlying relations from the data. In the next subsections, review of model based techniques for sequential recommendation such as (i) Matrix Factorization and (ii) Markov Models will be presented.

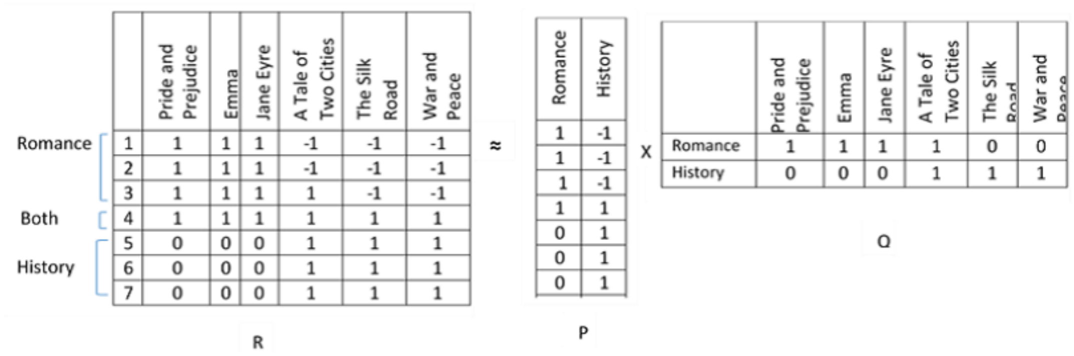
2.2.2.2.1 Matrix Factorization Matrix factorization (Koren, 2009) is used to discover latent features (concepts) underlying the interactions between two different kinds of entities such as users and items. The idea is to decompose the rating matrix into two smaller matrices in such a way that when multiplied together (by taking dot product), they approximate the original matrix and predict the unknown ratings. In the basic matrix factorization model, the mÖn ratings matrix R is approximately factorized into an $m \times k$ matrix P and an $n \times k$ matrix Q, as follows:

$$R \approx P \times Q^T = \hat{R} \tag{2.11}$$

Where m is the number of users, n is the number of items, k represents the number of latent features. Each column of P (or Q) is referred to as a latent vector or latent component (concepts), whereas each row of P (or Q) is referred to as a latent factor. The ith row p_i of P is referred to as a user factor, and it contains k entries corresponding to the association of user i towards the k concepts in the ratings matrix. Similarly, the ith row q_i of Q is referred to as item factor, and it contains k entries corresponding to the association of item i towards the k concepts in the ratings matrix. The concept behind matrix factorization is that there should be some latent features which determine the preference of a user towards an item. For example, two users may give high ratings to a certain book if they both like the writer of the book, or if the book is based on an historical event. In this case, the latent feature can be the "genre" which both users like. So, the latent feature "historic genre" can be used to describe both the user and the item. Thus, the discovery of such latent features can help to predict the rating of a particular user on a particular item because the features associated with both the user and the item are same. While determining those different features, an assumption is also made that the number of features (k) would be less than the number of users and the number of items. To get the prediction of a rating of an item q_j by u_i, we can calculate the dot product of the two vectors using Eq. 2.12.

$$\widehat{r_{lj}} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj} \tag{2.12}$$

Fig. 2.2 which shows a conceptual view of matrix factorization of the rating matrix



		Pride and Prejudice	Emma	Jane Eyre	A Tale of Two Cities	The Silk Road	War and Peace
Romance	1	1	1	1	-1	-1	-1
	2	1	1	1	-1	-1	-1
	3	1	1	1	1	-1	-1
Both	4	1	1	1	1	1	1
History	5	0	0	0	1	1	1
	6	0	0	0	1	1	1
	7	0	0	0	1	1	1

R

≈

Romance	History
1	-1
1	-1
1	-1
1	1
0	1
0	1
0	1

P

X

	Pride and Prejudice	Emma	Jane Eyre	A Tale of Two Cities	The Silk Road	War and Peace
Romance	1	1	1	1	0	0
History	0	0	0	1	1	1

Q

Figure 2.2: An example of rank-2 (k=2) Matrix Factorization (Ricci et al., 2015)

R into two matrices U and V. In the matrix U, ui is a 2-dimensional vector containing the association of user i towards the history and romance genres in the ratings matrix. Similarly, each row vi of V is referred to as an item factor, and it represents the association of the ith item towards these k concepts. In the matrix Q, the item factor contains the association of the item towards the two categories of books.

2.2.2.2.2 Markov Models Markov chain based models (Bernhard et al., 2016, Brafman et al., Rendle et al., 2010, Shani et al., 2005) are an early approach to Top-N sequential recommendation. Markov chain (MC) model based RS utilize sequential data by predicting the users' next action based on the last actions. The model detects sequential patterns through stochastic transitions between states. Therefore a transition matrix is estimated that gives the probability of transitioning to the next state based on the previous state (for example, buying an item based on the last purchase of the user). The transition matrix of the MC models is assumed to be the same over all users. An $L - order$ Markov chain makes recommendations based on L previous actions. The first order Markov chain is an item-to-item transition matrix learnt using maximum likelihood estimation (MLE).

Markov chain contains three components which are (i) the set of states (where a state can represent, an item, stock trends, weather conditions etc.,), (ii) the process function that directs transition from one state to another, and (iii) a start state s (Rubino and Sericola, 2014). For a given set of states $S = \{s_1, s_2, s_3, ..., s_n\}$, the process initiates from one of these states, and it moves successively one step towards another state. The probability of moving from a current state s_i to the next state s_j is denoted by probability $p(i, j)$. The probabilities are called transition probabilities. Therefore a transition matrix is estimated that gives the probability of transitioning to the next state based on the current state (for example, buying an item based on the last purchase of the user). In the transition matrix $\mathbf{P}$, the rows represent the current state, i.e., from s_i, the columns represent the next state, i.e., going to s_j. Each entry $p_{i,j}$ in the matrix is the conditional probability of going to the next state s_j, given the current state s_i, i.e., the probability of going from state i to state j (Rubino and Sericola, 2014). The transition matrix of the MC models is assumed to be the same over all users which is a draw back as it does not include an individual users' preference (personalization) into account while generating recommendations. Furthermore, it suffers from the problem of ambiguous prediction where two or more states (where a state is representing an item in this case) have the same transition probability from the current state (Li et al., 2019, Rubino and Sericola, 2014, Sarukkai, 2000). The process can remain in the state and this occurs with probability $p(i, i)$. An initial probability is assigned to the state which is designated as a start state.

Example 2.7. Next Item Recommendation using Markov Model

Consider an example to understand the process of Markov Model for next item recommendation by computing transition probabilities for transitions from current state to the next state.

Input: Finite set of states, $S = \{s_1, s_2, \ldots\ldots, s_n\}$, Initial Probabilities $\mathbf{p_s}$ for every s specifying the probability of starting from state s.

Output: Transition Matrix $\mathbf{P}^{t+1}$ with Transition Probabilities $P(s_j|s_i)$ defining the probability of going from state s_i to state s_j.

Figure 2.3 shows a state diagram and computation process to compute probability for transitioning to next state from current state (e.g., buying the next item after the purchase of a recent item) from time t to $t+1$ using first order Markov Model. Consider we have three states representing three products A, B and C and the transition probabilities of moving from one state to the next (probabilities of purchasing these products). Assume an initial transition matrix $\mathbf{P^t}$ with transition probabilities defining the probability of going from one state to another (based on analysis of customers' purchase data), in this case the probability of purchasing the next product. The column indicates the start state (s_i) and the rows indicate the end state (s_j). For example, at current time t, the probability of going from state A to B is 0.1 (probability of purchasing product B after product A). An initial probability vector $\mathbf{p_s}$ for each state to be the start state (chosen randomly) is also provided. For example, for the given vector $\mathbf{p_s}$ with values [0.40 0.24 0.36], the probability that the customer will start its purchasing journey from product A is 40% (0.40) (based on analysis from historic purchases on the company's site), whereas the probability that product B will be purchased first is 24% (0.24). To predict the transition probabilities from each of these states to next states at time $t+1$, we compute product of the transition matrix $\mathbf{P^t}$ with the initial probabilities vector $\mathbf{p_s}$. For example, as can be seen from the new probabilities $\mathbf{P^{t+1}}$ (at time $t+1$), the probability of purchasing product B after product A is 0.31 (second row in the probability transition matrix $\mathbf{P_{t+1}}$).

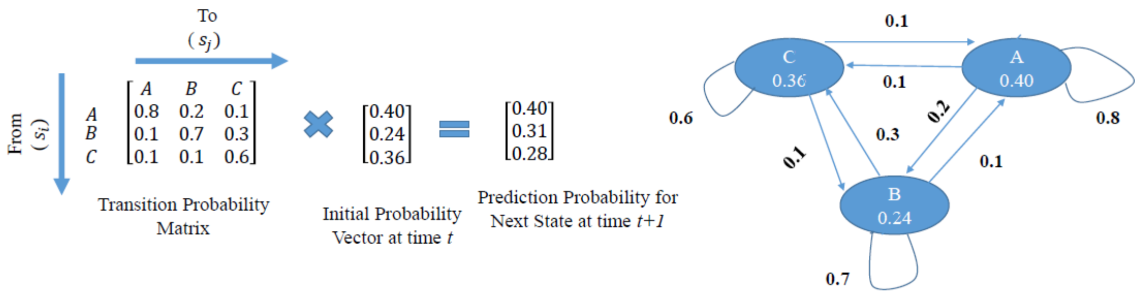

To
(s_j)
From
(s_i)
$\begin{array}{c|ccc} & A & B & C \\ A & 0.8 & 0.2 & 0.1 \\ B & 0.1 & 0.7 & 0.3 \\ C & 0.1 & 0.1 & 0.6 \end{array}$
$\begin{bmatrix}0.40\\0.24\\0.36\end{bmatrix}$
$\begin{bmatrix}0.40\\0.31\\0.28\end{bmatrix}$
Transition Probability Matrix
Initial Probability Vector at time t
Prediction Probability for Next State at time $t+1$
C 0.36
A 0.40
B 0.24
0.1
0.1
0.6
0.8
0.2
0.3
0.1
0.1
0.7

FIGURE 2.3: Example to compute probability for transitioning to next state from current state (e.g., buying an item) from time t to $t+1$ using first order Markov Model

The limitation of first order Markov model is that the next state prediction depends only on the current state. Using Markov models for prediction suffers from some limitations. An increase in the order of Markov Model also increases the number of states and the model complexity. Alternately, reducing the number of states leads to inaccurate transition probability matrix and limits the predictive power. To address this trade off, the All-*lth*-Order Markov model was proposed, such that if the *lth*-order Markov model cannot make the prediction then the $(l-1)th$-order Markov model is used. However, this model involves large number of states. Selective Markov models SMM (Deshpande and Karypis) that only store some of the states within the model, were proposed as a better solution to the mentioned trade off problem. However, this solution may fail when the data sets are very large.

Other works utilizing Markov Chains for sequential recommendation involve the work by Shani et al. (Shani et al., 2005) who introduced a recommender based on Markov decision processes (MDP) and also a MC based recommender. A sequential recommender based on Markov model is presented (Zimdars et al., 2013) to extract sequential patterns and learn the next state with a standard predictor, e.g., a decision tree. Rendle et al. (Rendle et al., 2010) proposed a factorized personalized Markov chains (FPMC) that outperforms traditional recommendation models by combining sequential information with user preference information. Despite the successful results of the FPMC, this model considers only the previous order information of the user, and the calculation is complicated due to the separation of the sequential pattern matrix and the user preference matrix. Similarly, (He and McAuley, 2016) presented a factorized sequential prediction with item similarity model (Fossil) which includes the advantages of the aforementioned approach. Besides, it adds the high-order Markov chains concept, and considers the item similarity model approach proposed by Kabbur et al. (Kabbur et al., 2013). Although Fossil surpassed the other previous methods, it only relies on the user's log information, so there is a limit to identifying similarity between items. Some other works (Li et al.) proposed to use a model based on topic-based hidden Markov Model to analyze temporal dynamics of users' preference which identifies the groups to which the users' belong and recommend what topic user will be mostly interested in reading. Scholarswalk, another Markov Model random-walk-based approach (Polyzou and Karypis, 2019) was proposed for generating sequential next course recommendation for students by capturing the sequential relationships between different courses.

In summary, Markov chain-based RS's have some limitations, they can only capture the short-term dependencies while ignoring long-term ones due to the Markov property which assumes that the current interaction depends on one or several most recent interactions only; on the other hand, they can only capture the point-wise dependencies while ignoring the collective dependencies over user-item interactions.

2.2.2.3 Neural Network Based

Neural networks (Nielsen, 2015) have natural strength to model and capture the comprehensive relations over different entities (e.g., users, items, interactions) in a sequence. Many deep neural networks for modeling sequential behavior have been studied including Recurrent Neural Networks (RNN), Convolutional Neural Networks and Graph Neural Networks. In the next subsections, we provide a review of neural network based sequential recommendation methods sub-classified as deep neural networks (Multi layer Perceptrons, Recurrent Neural Networks, convolutional Neural Networks, Graph Neural Networks) and advanced models (Attention Models, Memory Models, Mixture Models, Adversarial Networks).

2.2.2.3.1 Deep Neural Network Methods

Deep learning has its origins from the field of artificial neural networks (Silver et al., 2016). It deals with training multi-layer artificial neural networks. Deep learning is an evolving field in machine learning research (Zhu et al., 2017) and it aims to interpret data (e.g., sounds, images and texts) (Guo et al., 2017) by mimicking the function of human brain. Hinton et al. (Hinton et al., 2006) proposed the idea of deep learning in 2006. In deep learning, low level features are combined to generate a high level abstract representation of features (e.g., attribute categories). This high level feature representation facilitates the discovery of distributed representation of features in the data. ***Multi Layer Perceptrons-*** A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP). It has three layers, input layer, output layer and one or more hidden layer(s). The inclusion of more than one hidden layer in the architecture makes it a deep neural network. MLPs can also be interpreted as multiple layers stacked together performing non-linear transformations to learn hierarchical feature representations. A Multilayer perceptron begins with the input layer and forwards the data to the output layer after going through processing at the hidden layer(s). Depending on the output, the error (difference between the actual outcome and the predicted outcome) is computed and to minimize the error, it is back propagated to the model. The weights at each layer are adjusted after computing the derivate and the model is updated. This process is repeated until the ideal weights are learned and error is minimized (i.e., models' prediction accuracy improves). The number of layers and the number of neurons are the hyper-parameters of the model and can be tuned (to find ideal values) for optimized performance.

For next basket recommendation, NNrec (Wan et al., 2015) utilised neural network. It predicts a user's next basket by capturing the local context in user's last k baskets. The proposed model has four layers as: (1) input layer, which inputs a one hot vector encoding of user id and item id's present in user's recent k baskets, (2) embedding layer, which performs an average pooling operation (down-sampling the input by extracting important features) to product user's basket vector representations, (3) hidden layer, inputs the concatenated k basket vectors and user vector from the embedding layer and finally the (4) output layer, which predicts the probabilities (using a Softmax function) of each candidate item to be in user's next basket.

HRM (Wang et al., 2015) is another MLP based model for next basket recommendation. It aims to find correlations between users' long term and short term preferences. The model has two aggregation layers where each layer performs any of the max pooling or average pooling operation to extract important features from the input. The first layer finds item correlations in user's recent basket (short term preferences) by forming transaction level item vector representations and the second layer models users' general interests (long term preferences) by learning user representation among all users. The next items in the basket are then predicted on the basis of this learnt hybrid representations.

2.2.2.3.2 Recurrent Neural Networks Recurrent Neural Networks (RNNs) are type of neural networks that are suitable for sequence problems. Given a sequence of historical user-item interactions, an RNN-based RS tries to predict the next possible interaction by modelling the sequential dependencies over the given interactions. A basic RNN is a standard feed-forward Multilayer Perceptron Network (MLP) architecture (a network with single input, multiple hidden and single output layer) with addition of loops to the architecture. RNN's have internal memory which helps the model to store important things about the input and enables it to predict precisely about what is coming next. For this reason, RNN's are preferred for sequential data like time series, speech, text, financial data, audio, video and weather. They are designed to handle variable length sequence data. Figure 2.4 shows architecture of an RNN model.

A walk through example for next item prediction using RNN is provided in (Example 2.8).

Problem : Given a sequence of historical user-item interactions, predict the next possible interaction by modelling the sequential dependencies over the given interactions.

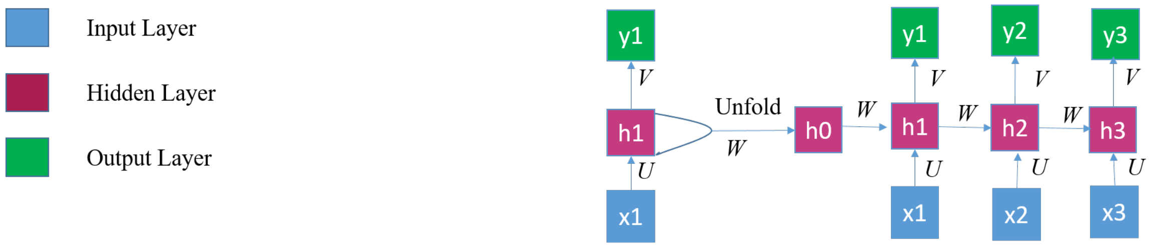


FIGURE 2.4: Architecture of an RNN Model

Example 2.8. Walk Through Example for next item prediction using RNN

A sample example is shown to explain the working of RNN based on (Hidasi et al., 2016c). A Gated Recurrent Unit (GRU)-based RNN for session-based recommendation (GRU4Rec) was proposed. The main contribution was to propose modification to GRU by introducing session parallel mini batches (to handle variable length sequences) and mini-batch based output sampling (to provide ranking on a subset of items). The input is the actual state of session with 1-of-N encoding, where N is the number of items. The coordinate will be 1 if the corresponding item is active in this session, otherwise 0. The output is the likelihood of the item being next in the session for each item. It starts by taking input of user click sessions and then creating an order for the sessions. The first event of each of the first X sessions forms the input of the first mini-batch. The second mini-batch is formed from the second events and so on. If any of the sessions end, the next available session is put in its place. The desired output is the second events of active sessions. Sessions are considered to be independent, thus the appropriate hidden state (GRU) is reset to zero when this switch occurs. A user click session consists of session id, item id and time stamps. Actual vocabulary size is in 100 and millions, but for the example, here, small vocabulary size of three items is used.

Input: User click sessions, hidden unit dimension=3, input layers=3, output layer=3, mini-batch size=3, weight vectors (U,V,W) as $U = \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix}$, $W = \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix}$, $V = \begin{bmatrix} -0.90 & 0.45 & 0.00 \\ 0.15 & 0.25 & 0.10 \end{bmatrix}$, item vocabulary = purse, wallet, mobile, laptop

Output: likelihood of item being next in the sequence

TABLE 2.17: User click sessions

Session No.	Click Sequences
S1	purse,wallet,laptop,mobile
S2	wallet,laptop,mobile
S3	mobile,purse,wallet,wallet,laptop,laptop
S4	mobile,purse
S5	wallet,laptop,mobile

TABLE 2.18: One-N hot vector encoding

Timestep	Input (Parallel -mini batches)	(1-N hot vector encoding)
1	(purse,wallet,mobile)	(1,0,0,0,1,0,0,0,1)
2	(wallet,laptop,purse)	(0,1,0,0,0,1,1,0,0)
3	(mobile,mobile,wallet)	(0,0,1,0,0,1,0,1,0)

Step 1: Parallel Mini batch creation

The mini batch will be created by taking the first event of each of the first X sessions to form the input of the first mini-batch (the desired output is the second events of active sessions). Table 2.17. shows sample click sessions. Here we use batch size =3. So, mbtch1=(purse,wallet,mobile), mbtch2= (wallet,laptop,purse), mbtch3= (mobile,mobile,wallet). Then create encodings as shown in Table 2.18, where presence of 1 indicates item position in the vector.

Step 2: Hidden layer Processing (Score calculation on items)

Pass the input word vector X_0 (input) at each timestep into the RNN cell to compute the next hidden state s_t using equation:

$$\text{st} = \sigma\left(x_t U + s_{t-1} W\right).$$

So, after substituting the values,we get the states as follows:

$$\text{s}_1 = \sigma \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} = \sigma[-0.75 \quad 0.50] = \begin{bmatrix} 0.32 & 0.62 \end{bmatrix}$$

$$\text{s}_2 = \sigma \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.32 & 0.62 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma \begin{bmatrix} 0.9 & 0.23 \end{bmatrix}$$

$$= \begin{bmatrix} 0.71 & 0.55 \end{bmatrix}$$

$$s_3 = \sigma \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.75 & 0.50 \\ -0.25 & 0.00 \\ 0.25 & 0.25 \end{bmatrix} + \begin{bmatrix} 0.71 & 0.55 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma \begin{bmatrix} 0.59 & 0.7 \end{bmatrix}$$

$$= \begin{bmatrix} 0.64 & 0.66 \end{bmatrix}$$

$$s_4 = \sigma \begin{bmatrix} 0.64 & 0.66 \end{bmatrix} * \begin{bmatrix} 0.10 & -0.25 \\ 0.50 & 0.50 \end{bmatrix} = \sigma \begin{bmatrix} 0.39 & 0.17 \end{bmatrix} = \begin{bmatrix} 0.59 & 0.54 \end{bmatrix}, \text{ similarly,}$$

$$s_5 = \begin{bmatrix} 0.57 & 0.52 \end{bmatrix},$$

$$s_6 = \begin{bmatrix} 0.57 & 0.52 \end{bmatrix}$$

Step 3: Prediction at timestep t

Calculating the output at each timestep using $o_t = softmax(V_{st})$, where V are the weights at output layer and s_t is the previously calculated hidden state.

$$o_4 = \text{softmax} \begin{bmatrix} 0.59 & 0.54 \end{bmatrix} * \begin{bmatrix} -0.90 & 0.45 & 0.00 \\ 0.15 & 0.25 & 0.10 \end{bmatrix} = \text{softmax} \begin{bmatrix} -0.45 & 0.40 & 0.05 \end{bmatrix}$$

Now using formula for calculating Softmax which is:

$S(y_i) = \frac{e^{yi}}{\sum e^{yj}}$, hence,

$S(-0.45) = \frac{e^{-0.45}}{e^{-0.45}+e^{0.40}+e^{0.05}}$,

similarly Softmax for 0.40 and 0.05 are also computed which gives us the output at this layer as o_4,

$$o_4 = \begin{bmatrix} 0.07 & 0.18 & 0.12 \end{bmatrix}, \text{ similarly,}$$

$$o_5 = \begin{bmatrix} 0.20 & 0.46 & 0.33 \end{bmatrix},$$

$$o_6 = \begin{bmatrix} 0.20 & 0.46 & 0.33 \end{bmatrix}$$

Step 4: Item Ranking From output o_4, o_5 and o_6, the results show likelihood for items to be next in the sequence.e.g., we have the output from the layers as:

Table 2.19: Scores and items' rank

Time step	Item Scores	Scores (descending)	Item Order	Expected order
1	(0.07,0.18,0.12)	(0.18,0.12,0.07)	Wallet,Mobile,Purse	Wallet , Mobile...
2	(0.20,0.46,0.33)	(0.46,0.33,0.20)	Wallet,Mobile,Purse	Laptop, Purse.....

$$o_4 = \begin{bmatrix} 0.07 \\ 0.18 \\ 0.12 \end{bmatrix}, o5 = \begin{bmatrix} 0.20 \\ 0.46 \\ 0.33 \end{bmatrix} o6 = \begin{bmatrix} 0.20 \\ 0.46 \\ 0.33 \end{bmatrix}$$

which means in o_4 out of three items "Purse, wallet and mobile", it ranked "wallet" as expected to be the next item (0.18) with high probability. In the next timestep t_4 and t_5 the output ranked wallet with the high score i.e. 0.46. wheraes we expect for items "laptop" and "mobile". However, the results may vary depending on the selection of weights. So, from this we can see that the model has ranked "wallet" as the next item in sequence for the next minibatch which is the same as the actual item in the next mini batch (next in sequence). The items are ordered according to their probability (descending order) and their position is their rank as shown in Table 2.19. If the recommended (predicted) item is not as the actual item, loss is computed and weights are updated by backpropagation.

Step 5: Loss Computation:

Expected loss is:

$$Ls = \frac{1}{Ns} \cdot \sum_{j=1}^{Ns} \sigma\left(r_{s,j-}r_{s,i}\right) + \sigma\left(r_{s,j}^2\right) \tag{2.13}$$

Where N_s is the number of samples in the session and $r_{s,i}$ is score on item i (next desired item) and $r_{s,j}$ are scores on negative samples which are taken from other sessions not present in the current session showing customer dislike. Hence,

$$\begin{aligned} \text{Ls} &= 1/3 * \sigma(0.46 - 0.18) + \sigma(0.46)^2 \\ &= 0.37 \end{aligned}$$

Similarly, loss for all sessions will be calculated for each epoch. After each epoch, the weight matrices U, W and V will be updated using Back Propagation Through Time (BPTT). Update to each weight matrix is proportional to the gradient of the error with respect to that matrix and BPTT computes the gradients using the chain rule. Figure 2.5 shows the computation steps by RNN architecture to generate the final recommendation for example 2.6.

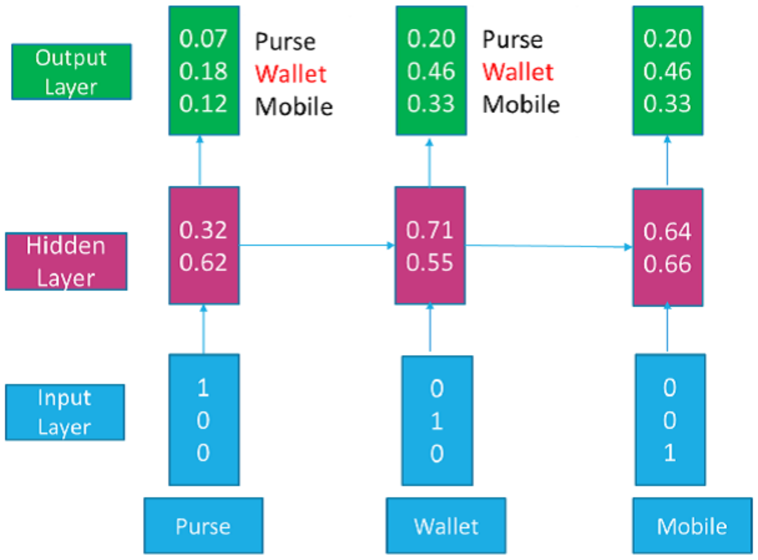


FIGURE 2.5: Recommended items (output) through RNN model for example 2.8

Some other variants of RNN's are proposed to capture more complex dependencies in a sequence, like hierarchical RNN (Quadrana et al., 2017). An extension to GRU4Rec (Hidasi et al., 2016c) was proposed (Tan et al., 2016) by providing four main features which are (1) click sequence augmentation, (2) temporal change adaptation, (3) privilege information usage and (4) using item embedding. Smirnova et al. (Smirnova and Vasile, 2017) proposed the use of contextual information besides considering the user's past sequence. They suggest using context such as the time gaps between events and the time of day for each interaction and the associated types of user-item interactions. Additionally, a new class of Contextual Recurrent Neural Networks for Recommendation (CRNNs) was proposed which incorporates the contextual information both in the input and output layers and modifies the behavior of the RNN by combining the context embedding with the item embedding.

One of the limitations of the previous session-based systems was that they were unable to capture evolution in user preferences across sessions. The reason being that those approaches handled sessions independently and there is no cross transfer of information across sessions. This issue was addressed through a hierarchical RNN (Quadrana et al., 2017). The model consists of a hierarchy of two GRUs, the session-level GRU (GRUses) and the user-level GRU (GRUusr). The session-level GRU models the user activity within sessions and generates recommendations. The user-level GRU(GRUusr) models the evolution of the user across sessions and provides personalization capabilities to the session-level GRU by initializing its hidden state and, optionally, by propagating the user representation in input. Another study (Hidasi et al., 2016c) introduced a parallel architecture consisting of multiple RNN's (p-RNN) for session-based recommendation. Three GRUs are used for learning item representations from text feature vectors, image features and one-hot vectors. To predict next item in the session, output of the three GRUs are weightedly concatenated and fed into a non-linear activation. They categorize the architecture as baseline RNN and Parallel RNN's. The idea is to first create multiple base line models each with a different item representation such as item ID, image and text. The proposed parallel architectures use a subset of the above three item representations (ID, image or text) as input. The output is a score for every item indicating the likelihood of being the next item in the session. The hidden states of these networks are merged to produce the score for all items. As a baseline, the standard best RNN setting from GRU4Rec15 was choosen. The input of the networks is the item ID of a transaction.

In a different study, a two stage GRU based RNN interactive recommendation model was introduced (Christakopoulou et al., 2018) which first asks users about topics of interest and then recommend items such as videos. This first stage is Question ranking. It involves capturing user interest by asking questions about the topic of their interest. The second stage is video response model (ranking stage model) which will predict Top-K videos based on the feedback (selected topic) from stage 1.

In summary, RNN's turns out to be promising models for modeling sequential dependencies between items, however they suffer from some limitations: (1) it is easy to generate fake dependencies due to the overly strong assumption that any adjacent interactions in a sequence must be dependent, which may not be the case in the real world because there are usually irrelevant or noisy interactions inside a sequence; and (2) it is likely to capture the point-wise dependencies only while ignoring the collective dependencies (e.g., several interactions collaboratively affect the next one), and (3) difficulty to train the model and optimize hyper parameters.

Convolutional Neural Networks- A Convolutional Neural Network (CNN) is a feed-forward neural network and consists of convolution layers, pooling layers, and feed-forward full-connected layers. It can efficiently capture the local and global features in the input. For example, common applications include processing time series and image data such as finding correlations between pixels in a specific part of an image or finding dependencies between various adjacent words in a sentence (Tang and Wang, 2018, Yuan et al., 2019).

For sequential recommendation, given a sequence of user-item interactions, a CNN first generates embeddings of these user-item interactions and creates a matrix of these embeddings. This matrix is then treated as an "image" in time and space. The CNN then uses convolutional filters to learn sequential patterns considering them as an image's local features and generates next recommendations. CNN's however, are not able to effectively capture long term dependencies due to limitation in the size of filters used. Some research works such as 3D-CNN (Tuan and Phuong, 2017) concatenated the embeddings of item id, name, and category to create the input embedding matrix for the CNN. CASER (Tang and Wang, 2018), another Convolutional Neural Network (CNN) based method provides sequential recommendation by applying convolution operations on the embedding matrix of L most recent items. It views the embedding matrix of L previous items as an 'image', and then captures the sequential patterns using horizontal and vertical convolutional layers. Long term preferences are also captured through user embedding. NextitNet (Yuan et al., 2019), a CNN model proposes to capture short and long term item dependencies through the use of residual block structure.

Graph Neural Networks- Recently, Graph Neural Networks (GNN) based systems have been adapted to sequential recommendation. They can capture complex relations and dependencies between user-item interactions in a sequence. These systems work by constructing a directed graph on the sequence data. Each interaction is considered as a node in the graph and each sequence gets mapped to a path in the graph. In the next step, user and item embeddings are learnt on the graph and complex relationships are embed into the graph (Wu et al., 2019). A Graph Neural Network based sequential recommender system (SDE-GNN) (Guo et al., 2021) captures sequential dependencies and item transition relations within sessions for generating accurate next item recommendations. Another recommender system (Xiao et al., 2020) generates unified friend and item recommendation by incorporating a mutualistic mechanism which models mutual relationships between consumption and social behaviour of users.

A Hierarchical and Interactive Gate Network (HIGnet) model for items' rating prediction is proposed (Zhong et al., 2020) which explores users' and items' textual features to capture their correlations by modeling informativeness of local words and captures global semantics from customer reviews in a hierarchical way. GNN (Zhou et al., 2020) is proposed to collectively aggregate information from graph structure. Wu et al. (Wang et al., 2018a) first used GNN for session-based recommendation by capturing more complex relationships between items in a sequence, and each session is represented as the composition of the long-term preference and short-term interests within a session using an attention network. Wu et al. (Wang et al., 2018a) fused GNN for session-based recommendation and proposed to model each session as a directed graph. Session representations include local and global session embeddings and the probabilities of next items are computed based on representations from sessions and item embeddings.

A deep listNet framework proposed by Wu et al. (Wu and Yan, 2017) with two phases as : (1) SIE, which performs pooling operation (average or max) to create click and view embedding vectors from user's interaction sequences. These are then concatenated with user and target item embedding. A feed forward network then learns a sessions' hybrid representation according to the concatenation. (2) List wise ranking, which computes relevance score between candidate items and session representations. These techniques capture high-order sequential interaction between items and users, however, they usually require a high computational time and are complex.

2.2.2.3.3 Advanced Deep Neural Networks

Attention Models- Attention mechanisms are also used for sequential recommendation. Given a user's historical behaviour (sequence), these models aim to identify items more relevant to the user. They focus more on interactions that are important in a sequence and ignore the ones that are less relevant. An encoder-decoder based attention framework NARM7 (Li et al., 2017) is proposed for transaction based sequential recommendation. The encoder is a hybrid of global and local encoder where the local encoder uses RNN with vanilla attention (which uses a linear combination of previous hidden vectors in the input sequence) to capture user's interest in a sequence and the global encoder uses RNN for modeling sequential dependencies between items in the current sequence. A unified representation for the sequence is then obtained from this hybrid encoder. Wang et al. (Wang et al., 2018b) also utilised vanilla mechanism and reduces impact of irrelevant interactions (e.g., accidental clicks, views) by assigning weights to each items. A short term attention model proposed by Li et al. (Li et al., 2017), computes (a) items' attention score in a sequence and (b) attention correlation between most recent and previous items in a sequence. A hierarchical attentive architecture (Bai et al., 2018) deploys attention mechanism at item feature level and identifies users' preferences towards items SASRec (Kang and McAuley, 2018a) uses attention mechanism to capture long term user preferences and uses user's historical sequence to identify preferences for next items. The model comprises of an embedding and self attention layer along with a feed forward network. The model can learn complex transitions between items by including additional attention layer and feed forward network (self attention blocks) in the architecture. The probability of a candidate item to be the next item (recommended next item in a sequence) is computed as a result of the output of the self attention block. BERT4Rec (Sun et al., 2019a), a Sequential Recommendation model with Bidirectional Encoder Representations from Transformer utilizes deep bidirectional self-attention mechanism for modeling user behavioral sequences and learns a bidirectional representation model which makes recommendations by allowing each item in users' historical behavior to integrate information from both left and right sides in given sequence. Zhang et al. (Zhang et al., 2018a) infers relationships between items by accepting an input of an item embedding vector, and predicts the next item by combining user's short term and long term preferences by estimating item weights in user's sequence using self attention and a metric learning framework respectively.

Memory Models- Memory Models utilise an external memory matrix to store and then later update the historical user-item interactions in a given sequence. This facilitates to find dependencies between historical and future user-item interactions dynamically and explicitly, hence improving model effectiveness and reducing interactions that are irrelevant (Chen et al., 2018). Some other works (Huang et al., 2018a) proposed to use a key-value memory network to store and then later update the historical user-item interactions to better learn user preferences for recommendation.

Mixture Models- A mixture model (Kang et al., 2018, Tang et al., 2019) generates sequential recommendation by integrating different sub models where each sub model captures different dependencies and contributes to the models' performance in generating better sequential recommendation. To learn short term and long term dependencies, (Tang et al., 2019) combines encoders of different kinds where each encoder learns a representation of a sequence to generate subsequent next recommendations.

Adversarial Networks- The Adversarial Network (AN) (Goodfellow et al., 2014) is a Generative Neural Network (GNN) which consists of a discriminator and a generator. The two neural networks (discriminator and generator) are simultaneously trained by competing with each other in a minimax game framework. Recently, generative adversarial networks (GANs) have been used in deep learning fields for image and audio generation (Zheng et al., 2019). GAN works on the principle of playing an adversarial minimax game between a generator and a discriminator. The task of the generator is to (i) capture distribution of real observed data, (ii) generate samples which are adversarial and (iii) fool the discriminator. The discriminator on the other hand focus on distinguishing whether the input sample is generated by the generator. Recommendation Systems DAN (Wang et al., 2019a) utilises an adversarial approach and transfers the representation of items and users to the target domain from different domains. Another GAN based model, AB-GAN (Zheng et al., 2019) a virtual try on clothing model which generates images for 2D images modeling. Using four features including user-image feature, desired-posture feature, body shape mask and new clothing feature, it generates an image of the user wearing new clothes. The generator creates the person's original image to ensure that data distribution is similar to the real image.

Some other Sequential Recommendation Systems (Nasir and Ezeife, 2020, Nasir et al., 2021) explored to integrate items' semantic knowledge obtained by utilizing item (products') meta data (e.g., title, description and brand) according to their semantic context (co-purchased and co-reviewed products) into the sequential recommendation process. The authors used Distributional hypothesis (Sahlgren, 2008) (which assumes that words which are similar in meaning occur in similar contexts), to learn an item's representation by analyzing the context in which they are used and then captures the semantics of these user-item interactions using techniques of Prod2Vec (Grbovic et al., 2015), Glove (Pennington et al., 2014), TF-IDF (Salton, 1988) and DoctoVec (Mikolov et al., 2013). The extracted items' semantic knowledge is then integrated into different recommendation phases such as pre-processing, candidate generation and the output (recommendation) phases.

A summary of Sequential Recommendation Systems as classified in the taxonomy based on the techniques used to build those systems along with their limitations and notable research works is presented in Tables 2.20, 2.21 and 2.22.

2.2.2.4 Classification of Traditional Sequential Recommendation Systems according to Features

2.2.2.4.1 Types of Sequences

Sequences in the database may represent information regarding single dimension or multiple dimensions. Below we will discuss both of these sequence types.

Single Dimensional Sequences- These are the sequences which represent only one aspect (dimension) of customers' transactions (for example, sequences representing products purchased by customers). Consider Table 2.23 which shows a single dimension sequence database which shows purchase sequences of four customers where SID represents the sequence ID to represent a unique customer and the sequence represents the items bought by the customer on different timestamps. For example, the customer with SID 100 has the purchase sequence $<(be)(ce)>$, representing two events, each comprising of two items. This shows that the customer first purchased items b and e together and then later on a different time stamp purchased items c and e together. Since, no other information besides the items purchased is represented by the sequence, these sequences are single dimensional. To mine frequent sequential purchase patterns according to a minimum support threshold from this single dimensional sequence database, any of the sequential pattern mining algorithms such as GSP (Agrawal and Srikant, 1995), SPAM (Ayres et al., 2002), Prefix Span (Jian Pei et al., 2001) and PLWAP (Ezeife et al., 2005) can be used as given the same sequence database and the minimum support threshold, any sequential pattern algorithm will yield same set of frequent sequential patterns and will only differ in the process of generating sequential patterns (i.e., the structure for extracting sequential patterns).

Table 2.20: Summary of Sequential Recommendation System Approaches

Category	**Sub-Category**	**Limitations**	**Research Work**
Traditional Approaches	Sequence Similarity Matching	· Static- Cannot learn user's evolving preferences · Item similarity computed using exact product match (e.g., name), no other content attribute information included · Sequence can only handle one event type (e.g., click/ purchase events) · No use of side information (users/items) · Do not capture temporal information	(Xiao and Ezeife, 2018), (Zhang and Cao, 2013), (Gurbanov and Ricci, 2017), (Song and Yang, 2014), (Salehi and Nakhai Kamalabadi, 2013)
	Sequential Pattern Mining	· Depends on choice of suitable threshold for min support · Unable to generate patterns that are less frequent(may be of interest to the target user) · Cannot model individual user's preferences (lack of personalization) · Cannot infer semantic relationships between items · Only recommending frequent (popular) products	(Bhatta et al., 2019), (Saini et al., 2017), (Choi et al., 2012), (Yap et al., 2012), (Rudin et al., 2011), (Parameswaran et al., 2010), (Liu et al., 2009), (Mobasher et al., 2002) (Agrawal and Srikant, 1995)
Factorization and Latent Representation	Factorization Techniques	. Latent Representations are not interpretable . Scalability issues . Good for only particular type of feedback (e.g., implicit user behavior)	(Zeng et al., 2019), (Pasricha and McAuley, 2018), (Wan and McAuley, 2018), (He and McAuley, 2016), (Lian et al., 2019), (Zhao et al., 2012) (Rendle et al., 2010)

TABLE 2.21: Summary of Sequential Recommendation System Approaches (continued from Table 2.20)

Category	**Sub-Category**	**Limitations**	**Research Work**
Factorization and Latent Representation	Markov Models	. Can only model short term dependencies (upto a limited no. of states) . Transition probability matrix suffers from ambigiuos prediction problem (same probability for transitioning to next state) and sparsity (lack of user-item interactions)	(Bernhard et al., 2016), (Tavakol and Brefeld, 2014), (Sahoo et al., 2012), (Shani et al., 2005), (Brafman et al.), (Deshpande and Karypis), (Zimdars et al., 2013)
Deep Neural Network Based	Multilayer Perceptrons	. Sensitive to feature scaling . Hyperparameter tuning (e.g., no. of layers, iterations)	(Xiao et al., 2020), (Dziugaite and Roy, 2015), (Lian et al., 2017), (Wang et al., 2017), (Xue et al., 2017), (Zhang et al., 2018b)
	Recurrent Neural Networks	. Assumes that adjacent interactions in a sequence must be dependent (which may not be true) . Difficult to train model and find optimal hyper parameters . Interpretability is low	(Hidasi et al., 2016a,b,c), (Villatel et al., 2018), (Donkers et al., 2017) (Wu et al., 2017), (Yu et al., 2016), (Christakopoulou et al., 2018), (Quadrana et al., 2017), (Tan et al., 2016)
	Convolutional Neural Networks	. Can be slower due to max-pooling operation . Difficulty in image classification if the image is tilted or rotated A possible solution is to use data augmentation during classification . Requires lots of training data	(Tang and Wang, 2018), (Tuan and Phuong, 2017), (Yuan et al., 2019), (Hsu et al., 2016)

TABLE 2.22: Summary of Sequential Recommendation System Approaches (Continued from Table 2.21)

Category	**Sub-Category**	**Limitations**	**Research Work**
Deep Neural Network Based	Graph Neural Networks	. Use of same parameters across different layers . Difficulty in effective modeling of informative features on graph edges	(Guo et al., 2021), (Xiao et al., 2020), (Wu et al., 2019), (Wu and Yan, 2017), (Sachdeva et al., 2019), (Zhong et al., 2020), (Zhou et al., 2020)
Advanced Deep Learning Models	Attention Models	. Addition of more weight parameters, hence increased model training time	(Wang et al., 2018b), (Ying et al., 2018), (Li et al., 2017), (Liu et al., 2018), (Ren et al., 2019), (Sachdeva et al., 2018), (Bai et al., 2018), (Kang and McAuley, 2018a), (Sun et al., 2019a), (Zhang et al., 2018a)
	Memory Networks	. An increase in each supporting memory level k, leads to an increase in the embedding matrix size according to models' vocabulary size which becomes impractical for huge vocabularies (e.g., with thousands of words)	(Chen et al., 2018), (Huang et al., 2018b)
	Mixture Models	. Identifiability (existence of a unique characterization for any one) of the models which requires minor constraints for estimation (e.g., in case of finite mixture models)	(Tang et al., 2019), (Wang et al., 2019b), (Kang et al., 2018)
	Adversarial Networks	. Mode collapse (occurs when the generator only produces a single output type or a small set of outputs) . Non-Convergence	(Goodfellow et al., 2014), (Unger et al., 2016), (Wang et al., 2018a)

Table 2.23: Single dimensional sequence data base

SID	Sequences
100	(be)(ce)
101	(ah)abf
102	(bf)(ce)(fg)
103	(bd)cba

Table 2.24: Multidimensional sequence database

SID	Cust_ type	City	Age_ range	Sequence
100	Student	Windsor	young	(be)(ce)
101	Manager	Toronto	middle	(ah)abf
102	Skilled Worker	Toronto	retired	(bf)(ce)(fg)
103	Student	Waterloo	young	(bd)cba

Multidimensional Sequences- In real world, most sequence patterns are associated with different circumstances and such circumstances form a multiple dimensional space. For example, customer purchase sequences are associated with location, time, customer group and occupation etc. Consider Table 2.24, which shows a multidimensional sequence database of customer purchases from Table 2.23 where the table now contains the columns (cust_type, city, age_range) representing three dimensions (aspects) of customers' information along with their unique SID's and purchase sequences. For example, for customer with SID 100, we can see that he is a student from the Windsor region and of young age. In other words, a multidimensional sequence will take the form as (a_i,a_2,.....,a_n,S), where a_i represents the dimension and S represents the sequence. In this case, for customer with SID 100, the multidimensional sequence will be represented as (student, Windsor, young, $<(be)(ce)>$). A multidimensional sequence is said to match a tuple in a multidimensional sequence data base if any of the dimensions in the multidimensional sequence matches with the corresponding dimension in the multidimensional sequence database and the sequence present in the multidimensional sequence is the subset of the sequence in the database. For example, a multidimensional sequence $M = (student, *, *, <(b)>)$ matches tuples 100 and 103 where $*$ in the multidimensional sequence represents any domain not present in the sequence database and the support (no. of matching tuples) of M in the sequence data base is 2.

Mining sequential patterns associated with multidimensional information are useful.

Table 2.25: Multidimensional Extension Sequence Database Created from Table 2.24

SID	Sequences
100	(student Windsor young)(be)(ce)
101	(manager Toronto middle)(ah)abf
102	(skilled worker Toronto retired)(bf)(ce)(fg)
103	(manager waterloo middle)(bd)cba

Given a multidimensional sequence database and a minimum support threshold, a multidimensional frequent sequential pattern will be the one whose support is greater than the minimum support threshold and can be mined through embedding multidimensional information into sequences and then mine the whole set using a sequential pattern mining method.

For example, consider Table 2.25 created as an extension from Table 2.24 by embedding the multidimensional information in the sequence as a special element. For instance, for tuple 100 in Table 2.24 $(100, student, Windsor, young, < (be, ce) >)$, the sequence $< (be)(ce) >$ in this tuple can be extended as $< (studentWindsoryoung)(be)(ce) >$ as shown in Table 2.25 (tuple 100). Other sequence can be extended in the similar fashion. Next, multidimensional sequential patterns can be mined from this extended sequence database by using any traditional sequential pattern mining algorithm such as Prefix Span (Jian Pei et al., 2001).

2.2.2.4.2 Use of side Information (Customers' & Items' meta data)

Traditional information sources consist of gathering users' interest on items by monitoring their implicit or explicit behaviors.

Implicit- Implicit actions are captured by inferring how a user responds to an item. For example, on an e-commerce site, which items a user has browsed, clicked, added to his favourite items' list, added to cart or purchased. These implicit actions indicate a user's interest on an item and are helpful in determining their future preferences. One way of expressing implicit behaviors is by storing binary values in the user item rating matrix where a "1" indicates behavior on an item such as clicked or purchased and a "0" otherwise. A disadvantage is that it cannot fully reflect users' preference on the item as a value of "0" does not necessarily mean that a user is not interested in the item. Since there are millions

Table 2.26: Historical Purchase Records of Customers

UID	Clicks sequence	Purchases sequence
1	(1,2,3),(7,5,3),(1,6),(6),(1,5)	(1,2),(3),(6),(7),(5)
2	(1,4),(6,3),(1,2),(1,2,5,6)	(1,4),(3),(2),(1,2,5,6)
3	(1,5),(6,5,2),(6),(5)	(1), (2),(6),(5)
4	(2,7),(6,6,7)	(2),(6, 7)

of products, it is very likely possible that the user is unaware of the existence of such items.

Explicit- Explicit behaviors are recorded when users' explicitly provide feedback on the item. A common example is rating an item on a five star scale where "1" is the lowest rating and "5" is the highest. Another way of extracting users' explicit feedback is from text reviews. In this case, if the review is positive, a value of "1" is stored in the user-item matrix, "0" for a neutral review and "-1" for a negative feedback. A common example is extracting user's preferences from the reviews they provide after consuming a product.

Auxiliary Input sources-These information sources complement the traditional information sources and can better reflect user's interest by capturing more insights about users' behavior. These information can be classified as (a) user and item information (e.g., meta data), (b) information contributed by users (e.g., tags, geotags, multimedia content, free comments and reviews) and (c) information associated with user-item interaction also known as context. An example of such information can be when a user is interacting with an item, such as purchasing an item, watching a movie or listening to a song (Adomavicius and Tuzhilin, 2011).

2.2.2.4.3 Consequential Bond between Click stream and Purchase Data

User interactions (clicks, purchases) in e-commerce data contains information which can be used to derive consequential bond (relationships) between these user interactions and thus facilitate in understanding user preferences (Xiao and Ezeife, 2018). More specifically, consequential bond refers to the concept that when a customer clicks on some items, it is most likely that some of those clicked items will be purchased. For example, consider Table 2.26, presenting historical purchase data of various customers. We can see that customer with user id 4 had clicked items 2 and 7 in a single transaction and notice that item 2 had been purchased in that transaction. Similar patterns can be seen in the behaviour of other customers which shows that there is a relationship between clicks and purchases and it can be used to derive the consequential bond between clicks and purchases.

2.2.2.4.4 Use of Contextual Information (location, occasion, season)

Unlike the traditional recommendation systems which collects users' preferences as ratings, context aware systems also include the "contextual information" to understand users' preferences. Context represents a set of factors or situations under which a user interacted with an item. For example, time, location, surroundings, purpose of purchase, device and occasion while interacting with an item (Adomavicius and Tuzhilin, 2011). Each context factor can be characterized by a structure such as the time factor can be described as seconds, minutes, hours, days, months and years. Such contextual information can be gathered from users' implicit or explicit feedback. For example, a user's rating for a hotel during her summer vacation stay.

2.2.2.4.5 Structure for Extracting and Modeling Sequential Behaviour

Different structures are used by Sequential recommendation Systems for extracting and modeling sequential behaviours. In traditional Sequential Recommendation Systems utilizing sequence similarity matching (Sect. 2.2.2.1.1), the technique of longest common sub-sequence is used to find the matching sequences for recommendation to the target user and for sequential pattern mining based approaches (Sect. 2.2.2.1.3), several algorithms are designed to mine sequential patterns from sequence databases such as GSP (Agrawal and Srikant, 1995), SPADE (Zaki, 2001), SPAM (Ayres et al., 2002), Prefix Span (Jian Pei et al., 2001) and PLWAP (Ezeife et al., 2005). The input to these sequential pattern mining algorithms is a sequence database and a minimum support threshold (set by the user) and output is the set of frequent sequential patterns. Given the same sequence data base and the minimum support threshold, all sequential pattern mining algorithms will output the same set of sequential patterns as these algorithms do not differ in the output but differ from each other according to the (1) type of search technique used, such as breadth first, e.g., GSP (Agrawal and Srikant, 1995), depth first, e.g., SPADE (Zaki, 2001) and SPAM (Ayres et al., 2002), (2) the type of database representation, such as, horizontal (e.g., GSP (Agrawal and Srikant, 1995) or vertical (e.g., SPADE (Zaki, 2001) and SPAM (Ayres et al., 2002) and (3) how next patterns are determined and generated from the search space, such as, candidate generate and test, e.g., Apriori (Agrawal et al.) and GSP (Agrawal and Srikant, 1995), pattern growth by using projected databases, e.g., Prefix Span (Jian Pei et al., 2001). Use of efficient strategies and data structures results in some algorithms more efficient than others (Fournier-Viger and Lin, Mabroukeh and Ezeife, 2010). For more details on structures for extracting sequential patterns using sequential pattern mining algorithms, we encourage the reader to follow the survey on sequential pattern mining by (Fournier-Viger and Lin, Mabroukeh and Ezeife, 2010, Mooney and Roddick, 2013).

Factorization and Latent Representation based approaches model user's sequential behaviour by creating a model from the training data (observed user ratings) and then predicting the ratings for users on unseen items based on the observations learned from the trained model. Model-based methods apply machine learning techniques (e.g., Matrix Factorization) during the model training process. Model-based methods are assumed to capture underlying relations from the data. Common approaches are Matrix Factorization (Koren, 2009) and Markov Models (Bernhard et al., 2016, Brafman et al., Rendle et al., 2010, Shani et al., 2005). The concept behind matrix factorization is that there should be some latent (hidden) features which determine the preference of a user towards an item. For example, two users may give high ratings to a certain book if they both like the writer of the book, or if the book is based on an historical event. In this case, the latent feature can be the "genre" which both users like. So, the latent feature "historic genre" can be used to describe both the user and the item. Thus, the discovery of such latent features can help to predict the rating of a particular user on a particular item because the features associated with both the user and the item are same. While determining those different features, an assumption is also made that the number of features k would be less than the number of users and the number of items. The Markov chain based models (Bernhard et al., 2016, Brafman et al., Rendle et al., 2010, Shani et al., 2005) are successful probabilistic models that model sequential patterns by predicting user's future actions based on the past actions. To achieve this goal, an item to item transition probability matrix is estimated where items which are nearest (with high probability of transitioning) after the recently interacted item by the user are recommended. For example, recommending the item for purchase that has the highest transitioning probability after the last purchased item by the user as explained in Sect. 2.2.2.2.1.

Deep Neural network based approaches (Nielsen, 2015) model user's sequential behaviour capturing the comprehensive relations over different entities (e.g., users, items, interactions) in a sequence. The most commonly used neural networks for modeling sequential behavior are Recurrent Neural Networks (RNN) (Christakopoulou et al., 2018, Hidasi et al., 2016b,b, Quadrana et al., 2017, Tan et al., 2016) due to their natural strength in sequence modelling.

Given a sequence of historical user-item interactions, an RNN-based RS tries to predict the next possible interaction by modelling the sequential dependencies over the given interactions. A basic RNN is a standard feed-forward Multilayer Perceptron Network (MLP) architecture (a network with single input, multiple hidden and single output layer) with addition of loops to the architecture. RNN's have internal memory which helps the model to store important things about the input and enables it to predict precisely about what is coming next. For this reason, RNN's are preferred for sequential data like time series, speech, text, financial data, audio, video and weather. They are designed to handle variable length sequence data. Several sequential recommender systems based on RNN's had been proposed to model users' sequential behaviour (Christakopoulou et al., 2018, Hidasi et al., 2016b,c, Quadrana et al., 2017, Smirnova and Vasile, 2017, Tan et al., 2016, Villatel et al., 2018).

2.2.2.4.6 Sequences having Long User Item Interactions

A long user-item interaction sequence has large number of user-item interactions which makes it challenging to model comprehensive and complex dependencies between multiple interactions that form the sequence. Two key features in long user-item interaction sequences are (i) learning higher-order sequential dependencies and (ii) learning long-term sequential dependencies.

Higher order sequential dependencies- Compared to the lower-order sequential dependencies, which are moderately simple and can be easily modeled by Markov chain models (Bernhard et al., 2016, Brafman et al., Rendle et al., 2010, Shani et al., 2005) or factorization machines (Koren, 2009), higher-order sequential dependencies are more complex and difficult to capture due to their complicated multi-level cascading dependencies crossing multiple user-item interactions. High order sequential dependencies can be addressed through the use of higher-order Markov-chain models (He and McAuley, 2016) and Recurrent Neural Networks (Christakopoulou et al., 2018, Hidasi et al., 2016b,c, Quadrana et al., 2017, Smirnova and Vasile, 2017, Tan et al., 2016). However, each approach has its own limitations, for example, an increase in the order of Markov Model also increases the number of states and the model complexity. Alternately, reducing the number of states leads to inaccurate transition probability matrix and limits the predictive power whereas in case of RNN models, some sequential dependencies may not be modelled efficiently due to the assumption that any adjacent items in a sequence are highly dependent, which might not be true in the real world as a sequence may contain irrelevant or noisy interactions.

Long-term sequential dependencies- In a sequence, long term dependencies involve dependencies between interactions which are far from each other. For example, in a given shopping sequence $S = \{towel, eggs, bread, butter, soap\}$, which consists of items purchased successively by a user Smith. Obviously, the towel and the soap are highly dependent (in the context of being used for bath) besides even they are far from each other. These situations are quite common in real world as users' behaviours tend to change with time and highly uncertain which may lead to placement of any items in any order in the cart. To address this, Long Short Term Memory based (Wu et al., 2017) and Gated Recurrent Unit based RNN (Hidasi et al., 2016c) have been used in Sequential Recommendation Systems to capture the long-term dependencies among the user-item interactions in a sequence. However, it is easy for RNN models to generate false dependencies by overly assuming any adjacent items in a sequence are highly dependent. In the above example of Smith's shopping sequence, an RNN usually models S by assuming the butter and the soap are dependent due to the close distance between them, but actually they are not. Overall, the works that are able to tackle this are quite limited and need further investigation.

2.2.2.4.7 Sequences with Flexible Order

In real world, it is not necessary that all adjacent user interactions in a sequence are dependent sequentially. For instance, in a shopping sequence $S = \{butter, eggs, cheese, bread\}$, the order of purchase of bread, eggs and cheese is not important, however, the collective purchase of *butter*, *eggs* and *cheese* leads to an increased probability to purchase *bread*. This shows that while there is no strict order among *butter*, *eggs* and *cheese* but the purchase of *bread* will depend sequentially on their union. Hence, in a flexible order sequence, it is important to capture collective sequential dependencies instead of point wise dependencies (which do not consider a strict order) between user-item interactions. Existing Sequential Recommendation Systems designed based on Markov-chains, factorization machines or RNN's are good at handling the point-wise dependencies but are not very good at modelling and capturing collective dependencies.

2.2.2.4.8 Temporal Patterns

Temporal databases integrate the concept of time to capture past, present and future data. Various forms of temporal data can include (i) start time (when the event actually began) and end time (when the event ended), for example, when a customer started his session by navigating an on line e-commerce site and ended the session after purchase or without a purchase by leaving the site, (ii) granularity (for example, events occurring on the same day or happening with N weeks from a specific day or in a specific month etc.) and (iii) periodicity which refers to regular repetition of a certain event within a specific time interval. For instance, the event “fall” is repeated once within the time interval of a year. Such events are called periodic events and such intervals are called periodic intervals (Alkilany, 2013).

Data is collected in the form of event time sequences where each event lasts for a certain time interval. Each record in the database stores the start time and end time during which the tuple is valid. For instance, for an e-commerce business, customers' purchase records are stored to determine customers' purchase behaviour over a certain period of time. For example, records like “Customer A had purchased items X and Y on 20 August” or Customer A visited the site from 13:00 to 14:00 on May 11 and purchased items X, Y and Z are stored. The temporal nature of data provides valuable information about varying trends or patterns. For example, we can find patterns like “80% of the customers who bought item X and then bought item Y within an hour, are likely to visit the site the following day to view new promotions”. Such frequent temporal patterns from customers are useful to identify correlations between items for further marketing and promotion strategies.

Incorporating temporal aspect is an important extension as it offers the capability to infer causal and temporal proximity relationships. The time component helps in analyzing the changes in data overtime. The time component may facilitate in identifying the validity of rules like $HikingGear \rightarrow HikingBoots$, $Years : Months(5 : 3)$ during $\{Years(2015), Years(2020)\}$ which reveals that every spring time from 2015 to 2020, customers who bought hiking gear also purchased hiking boots. Such a rule may not be valid before 2015 or after 2020. Therefore, by adding the temporal information to the rule set, more accurate and clear information is obtained. Furthermore, it is possible to predict how quickly a domain changes by discovering the change in knowledge obtained from the underlying data, thus leading to better marketing strategies.

A classification of sequential recommender systems is presented in Tables 2.27, 2.28 and 2.29 according to the categories presented in the taxonomy and based on the eight features explained above.

2.2.3 Semantics Based Recommendation Techniques

Research works have introduced to learn item associations by learning their semantics (meanings). This is achieved by transitioning from key-word based representations to concept-based user and item representations. Availability of open knowledge sources such as Wikipedia[1] (a free online encyclopedia), DBpedia[2] (structured representation of content extracted from Wikipedia) and BabelNet[3] (a multilingual encyclopedic dictionary) have largely contributed in the analysis of the content by integrating techniques from semantic technologies such as ontologies (formal representations of categories, properties and relations between concepts, data and entities), Resource Description Framework (framework for describing resources on the web) and Natural Language Processing (NLP), leading to

[1]www.wikipedia.org
[2]www.dbpedia.org
[3]www.babelnet.org

Table 2.27: Classification of Sequential Recommender Systems According to Features

	Sequences Type	Side Information (Meta data) (Users, items)	Consequential Bond	Contextual Information	Structure to Extract Sequential Patterns	Handling Long user Sequences	Flexible Order Sequences	Temporal Characteristics (Sequence Granularity)
ChoiRec12 (Choi, K., Yoo, D., Kim, G., & Suh, Y. , 2012)	Single level	x	x	x	Sequential Pattern Analysis	x	x	x
(Yap, G. E., Li, X. L., Philip, S. Y., 2012)	Multi-level	x	x	x	Pattern growth (Prefix Span)	x	x	Yes (weighted Backward and Forward Compatibility of sequences with the target sequence)
Hybrid (Salehi, M. , 2013)	Single level	Yes (Product Attributes)	x	x	Apriori (introduced weighted association rules)	x	x	Yes (more weight assigned to products purchased recently)
BSSM (Zhang, Y., & Cao, J., 2013)	Single level	x	x	x	Sequence similarity based on behavior Sequence Similarity	x	x	x
(Wei Song Kai Yang, 2014)	Single level	x	x	x	Weighted Sequence Matching	x	x	x

Table 2.28: Classification of Sequential Recommender Systems According to Features (Continued from Table 2.27)

	Sequences Type	Side Information (Meta data) (Users, items)	Consequential Bond	Contextual Information	Structure to Extract Sequential Patterns	Handling Long user Sequences	Flexible Order Sequences	Temporal Characteristics (Sequence Granularity)
SainiRec17 (Saini, S., Saumya, S., & Singh, J. P. , 2017)	Single level	x	x	x	Vertical Id list based (SPADE)	x	x	Yes (monthly basis)
Hybrid (Gurbanov, T., & Ricci, F. (2017)	Single level, however (with collection of various actions on the event, like click, bookmark etc)	x	x	x	Probabilistic (Bayes and Naïve Bayes)	x	x	Yes (Sessions as time less and time aware) Where in time aware represent the time gap between action a_i and action a_{i+1} on the same item
HPCRec18 (Xiao, Y., & Ezeife, C. I. , 2018)	Single-level	x	Yes (without considering the sequentiality)	x	Consequential Bond	x	x	x

Note: X means the presence of a particular feature in the system under discussion

Table 2.29: Classification of Sequential Recommender Systems According to Features (Continued from Table 2.28)

	Sequences Type	Side Information (Meta data) (Users, items)	Consequential Bond	Contextual Information	Structure to Extract Sequential Patterns	Handling Long user Sequences	Flexible Order Sequences	Temporal Characteristics (Sequence Granularity)
HSPRec19 (Bhatta et al., 2019)	Single-level	x	Yes	x	GSP (Generalized Sequential Patterns)	x	x	Yes (Creating Sequences with granularity of daily, weekly and monthly purchases)
SEMSRec20 (Nasir, M., Ezeife, C. I., 2020)	Single-level	Yes (customers' reviews)	Yes	x	Pattern growth (Prefix Span), Use of semantics during mining	x	x	Yes (Creating Sequences with granularity of daily, weekly and monthly purchases)
SEMSRec21 (Nasir, M., Ezeife, C. I., Gidado, Abdulrauf,2021)	Single-level	Yes (customers' reviews & product meta data)	Yes	x	Pattern growth (PrefixSpan) Use of semantics during mining	Yes	x	Yes (Creating Sequences with granularity of daily, weekly and monthly purchases)

Note: X means the presence of a particular feature in the system under discussion

semantic recommenders. According to (Lops et al., 2011), techniques in semantic recommenders can be categorized as (i) top down and (ii) bottom-up.
To capture the semantics of target user's information needs, ***top-down approaches*** utilize external knowledge sources, such as, taxonomies (IS-A hierarchy), dictionaries, or ontologies for creating user profiles and interpreting the meaning of items. Top down approaches aim to facilitate recommender systems in interpreting documents written in natural language and provide meaningful reasoning by providing knowledge such as linguistic, common sense and cultural backgrounds.

Bottom up approaches, on the other hand, interpret the semantics by exploring the syntagmatic and paradigmatic relations between words in high dimensional vector spaces. Syntagmatic relation is a type of semantic relation between words that co-occur in the same sentence or text (Asher and Simpson, 1993). For example, "the lion chased the deer", representing syntagmatic relationship between lion and chase or "I liked my new iphone" representing relationship between "iphone" and "liked". Paradigmatic relation is a different type of semantic relations between words that can be substituted with another word in the same categories for example, substituting 'cat' with 'dog' or 'coca-cola' with 'coca-cola cherry'. Bottom up approaches are also called distributional approaches, as they learn a representation by analyzing the context (neighborhood) in which the word is used. The idea is that words (or terms) that co-occur in a context are likely to be similar to each other and therefore more close in the vector space.

2.2.3.1 Top Down Semantic Approaches

These approaches are based on the integration of external knowledge for representing content. They facilitate the recommendation system by providing the linguistic, cultural and background knowledge in the content representation. They introduce semantics by (i) mapping the features describing the item with semantic concepts, such as Word Sense Disambiguation (Semeraro et al., 2007), entity linking or (ii) linking the item to a knowledge graph such as ontological knowledge (Middleton et al., 2004), structured or unstructured encyclopedic knowledge, like Wikipedia (Gabrilovich and Markovitch, 2009, Semeraro et al., 2009) or utilizing Linked Open Data Cloud (Di Noia et al., 2012). In the next subsection each of these methods for inferring semantics from the content will be discussed.

Table 2.30: WordNet structure

Word Meanings						Word Forms
	F1	F2	F3	...	...	Fn
M1	V(1,1)	V(2,1)				
M2		V(2,2)	V(3,2)			
M3						
M...						
M..n						
						V(m,n)

2.2.3.1.1 Word Sense Disambiguation WSD (Semeraro et al., 2007) selects the proper meaning, that is, sense, for a word in a text by taking into account the context in which it occurs. The main idea is to learn the semantics by utilizing linguistic ontology such as WordNet (Miller, 1995), which is a lexical database for the English language. WordNet is used to integrate linguistic knowledge while learning user profile. WordNet arranges words into sets of synonyms called synsets, which represent specific meaning of a word along with its synonyms. Therefore, items are represented according to the synsets that are likely to indicate their characteristics and user profiles are created according to the synsets that specify their preferences. Table 2.30. shows a sample structure for WordNet where each row represents synonyms and each column represents polysemous words (same word with multiple meanings).

JIGSAW Algorithm in WSD:

The JIGSAW algorithm (Basile et al., 2007) is a Word Sense Disambiguation (WSD) system which disambiguates (represent with semantics) all words in a text based on exploiting WordNet senses. It disambiguates (represents with semantics) each word w_i in the document according to the context of each word (synset).

Problem : Given a collection of words in documents, represent each document as semantics (by disambiguating the words) in k dimensions where $k <$ number of words. The input to the algorithm is a document $D = \{w_1, w_2, \ldots, w_h\}$ where w_1, w_2, represents words in the documents, the algorithm outputs the document as a representation of semantics in k dimensions such that $X = \{s_1, s_2, \ldots\ldots, s_k\}$ $(k < h)$, where each s_i is obtained by disambiguating w_i according to the context of each word. It uses semantic similarity between synsets using path length similarity such as Leackock-Chodorow similarity (Leacock and Chodorow, 1998).

Example 2.7. Synset Semantic Similarity Computation (Taxonomy based) Consider an example of computing the synset semantic similarity between two terms "cat" and "mouse" according to a pre-defined taxonomy as shown in Fig. 2.6

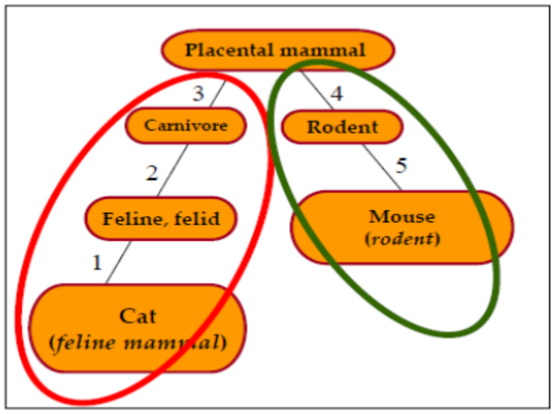


FIGURE 2.6: A sample taxonomy of animals (Mammal)

The similarity is computed by using Eq. 2.14,

$$\text{SINSIM} = r = -\log\left(\frac{N_p}{2D}\right) \tag{2.14}$$

where N_p is the number of nodes in path p from a to b (where a = cat and b = mouse) and D is the maximum depth of the taxonomy. The value of D in WordNet is 16 and according to the taxonomy (IS-A) relationship, value of N_p is =5. Hence, semantic similarity between the two terms *cat* and *mouse* is,

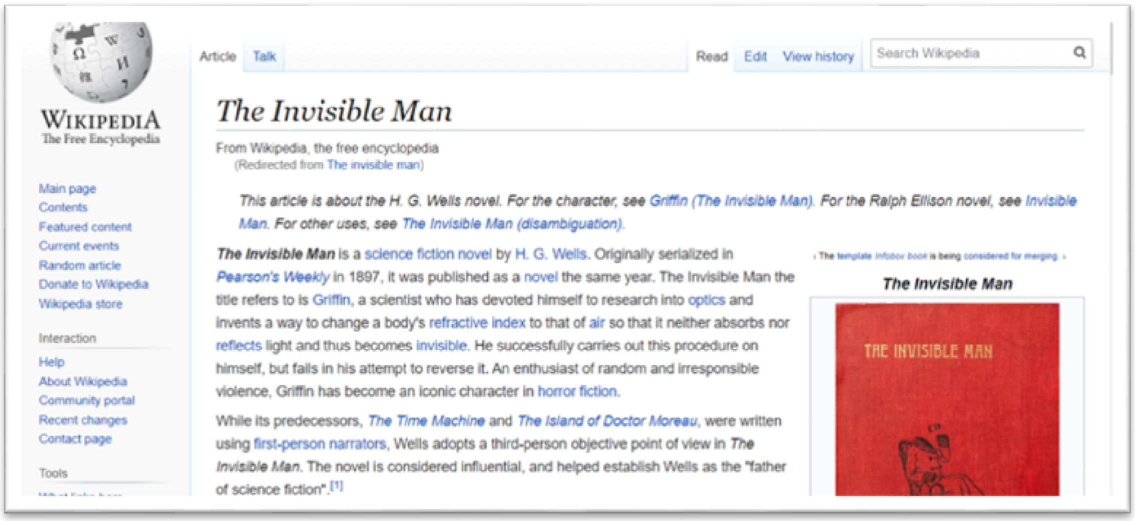
$$\text{SINSIM (cat, mouse)} \; = -\log\left(\mathrm{N_p/2D}\right) = 5/2 * 16 = 5/32 = 0.86.$$

Example 2.8. Synset Semantic Similarity Computation (Key word based) Consider the example where each book is represented as a set of keywords. For example, the book "The invisible Man" was under the "Science Fiction" genre and represented as a set of key words such as science fiction, novel, 1897, horror, H.G Wells. Through WSD, we can process the textual description of the item can obtain a semantics-aware representation

Table 2.31: Synsets obtained from WordNet

Keyword	Synsets obtained from WordNet
Science fiction	{06380251} S: (n) science fiction (literary fantasy involving the imagined impact of science on society)
Horror	{07535650} S: (n) horror (intense and profound fear)
	{03543047} S: (n) horror (something that inspires horror; something horrible)
	{07519226} S: (n) repugnance, repulsion, revulsion, horror (intense aversion)
Invisible	{02527322} S: (adj) invisible, unseeable (impossible or nearly impossible to see; imperceptible by the eye)

of the item as output, that is, keyword-based features are replaced with the concepts from WordNet synsets. A Wikipedia page for the book is shown in Fig. 2.7 A sample of key-



The Invisible Man

From Wikipedia, the free encyclopedia
(Redirected from The invisible man)

This article is about the H. G. Wells novel. For the character, see Griffin (The Invisible Man). For the Ralph Ellison novel, see Invisible Man. For other uses, see The Invisible Man (disambiguation).

The Invisible Man is a science fiction novel by H. G. Wells. Originally serialized in *Pearson's Weekly* in 1897, it was published as a novel the same year. The Invisible Man the title refers to is Griffin, a scientist who has devoted himself to research into optics and invents a way to change a body's refractive index to that of air so that it neither absorbs nor reflects light and thus becomes invisible. He successfully carries out this procedure on himself, but fails in his attempt to reverse it. An enthusiast of random and irresponsible violence, Griffin has become an iconic character in horror fiction.

While its predecessors, *The Time Machine* and *The Island of Doctor Moreau*, were written using first-person narrators, Wells adopts a third-person objective point of view in *The Invisible Man*. The novel is considered influential, and helped establish Wells as the "father of science fiction".[1]

‹ The template Infobox book is being considered for merging. ›

The Invisible Man

Figure 2.7: Wikipedia page for the book "The invisible Man"

word replacement to concepts based on synsets obtained from WordNet for the book "The Invisible Man" are shown in Table 2.31.

The format shows the synset ID in the word net as in curly brackets {} and then the concepts associated with the keyword.

WSD can handle the polysemy and synonymy. However, the limitation of WSD is that it cannot recognize the entities such as people, places and organizations in the text. For example, H.G. Wells in the content description of the book.

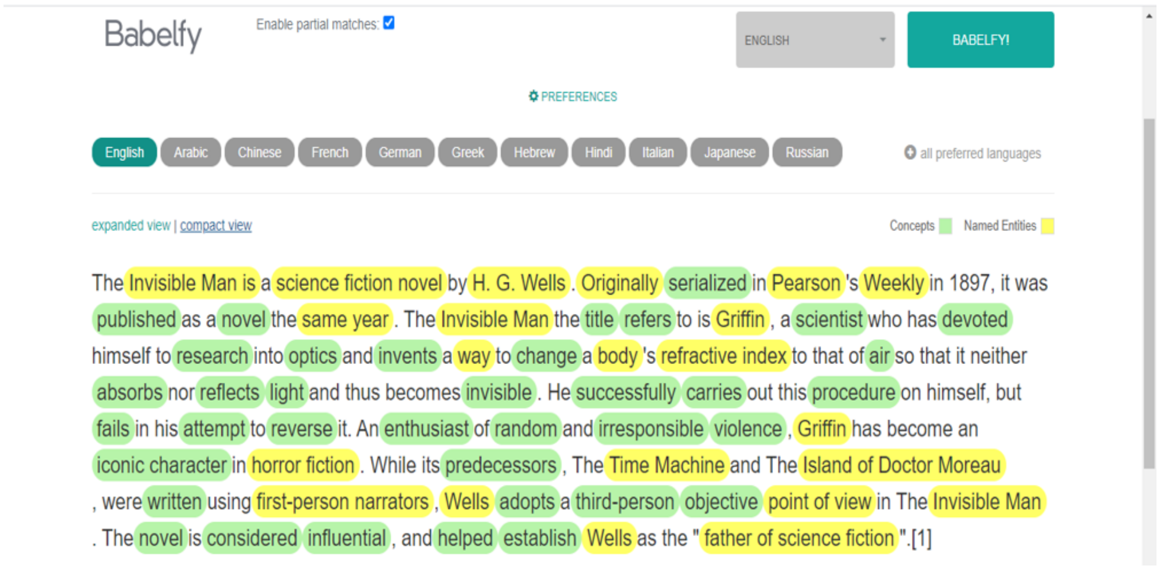
2.2.3.1.2 Entity Linking Entity linking algorithms address the problem of entity recognition in the content. Input to these algorithms is free text such as content description from Wikipedia and output is the identification of entities in the text. Entity linking is important because we need to identify the entities mentioned in the textual description to better catch user preferences and information needs. Several state-of-the-art implementations for entity identification from text are already available namely OPENCLAIS[4] , TAG.ME[5] and Babelfy[6] . OPENCALAIS can correctly identify entities in the text, but it ignores common sense and abstract concepts. TAG.ME is very transparent and provides human-readable content representation. It annotates the text with relevant hyperlinks to the Wikipedia pages. Non-trivial NLP tasks, such as, stop words removal, n-grams identification, named entities recognition and disambiguation are automatically performed. The advantage of extracting entities and concepts from text using TAG.ME are (i) several common sense concepts are now identified like (novel, science, optics) and each identified entity is a reference to a Wikipedia page. For example, the entity H. G. Wells will refer to the Wikipedia page `https://en.wikipedia.org/wiki/H._G._Wells.`

By utilizing this information, we can enrich this entity-based representation by exploiting the Wikipedia categories' tree. The enriched representation can be created by combining the entities and concepts identified in the text along with the most relevant Wikipedia categories each entity is linked to. However, the limitation is that it ignores terms and entities which cannot be linked to Wikipedia. Babelfy is a large multilingual encyclopedic dictionary and semantic network that uses graph-based approach to Entity Linking and Word Sense Disambiguation.

Babelfy is also good at identifying entities and concepts from text and by linking entities to Wikipedia-based representation, it can extract some common sense terms and new interesting features (for example, science-fiction writer) can be generated. It also includes linguistic knowledge and is able to disambiguate terms. Fig.2.8 shows entities identified from the content description of the book "The Invisible Man". The yellow highlighted words shows the entities and green ones shows the concepts. Fig. 2.9 shows entities iden-

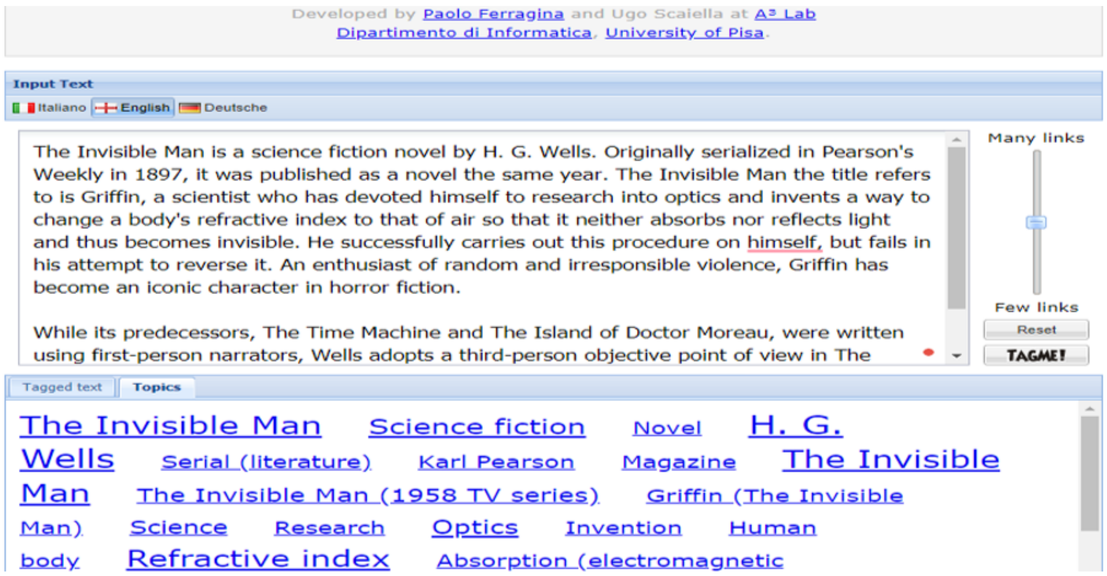
[4]http://www.opencalais.com/opencalais-api/
[5]https://tagme.d4science.org/tagme/
[6]http://babelfy.org/

tified from the same text using TAG.ME. Underlying entities shows their links to relevant Wikipedia pages.



The Invisible Man is a science fiction novel by H. G. Wells . Originally serialized in Pearson 's Weekly in 1897, it was published as a novel the same year . The Invisible Man the title refers to is Griffin , a scientist who has devoted himself to research into optics and invents a way to change a body 's refractive index to that of air so that it neither absorbs nor reflects light and thus becomes invisible . He successfully carries out this procedure on himself, but fails in his attempt to reverse it. An enthusiast of random and irresponsible violence , Griffin has become an iconic character in horror fiction . While its predecessors , The Time Machine and The Island of Doctor Moreau , were written using first-person narrators , Wells adopts a third-person objective point of view in The Invisible Man . The novel is considered influential , and helped establish Wells as the " father of science fiction ".[1]

FIGURE 2.8: Text to entity and concept identification through Babelfy


Developed by Paolo Ferragina and Ugo Scaiella at A³ Lab
Dipartimento di Informatica, University of Pisa.
Input Text
Italiano
English
Deutsche
Many links
Few links
Reset
TAGME!
Tagged text
Topics

The Invisible Man is a science fiction novel by H. G. Wells. Originally serialized in Pearson's Weekly in 1897, it was published as a novel the same year. The Invisible Man the title refers to is Griffin, a scientist who has devoted himself to research into optics and invents a way to change a body's refractive index to that of air so that it neither absorbs nor reflects light and thus becomes invisible. He successfully carries out this procedure on himself, but fails in his attempt to reverse it. An enthusiast of random and irresponsible violence, Griffin has become an iconic character in horror fiction.

While its predecessors, The Time Machine and The Island of Doctor Moreau, were written using first-person narrators, Wells adopts a third-person objective point of view in The

The Invisible Man Science fiction Novel H. G. Wells Serial (literature) Karl Pearson Magazine The Invisible Man The Invisible Man (1958 TV series) Griffin (The Invisible Man) Science Research Optics Invention Human body Refractive index Absorption (electromagnetic

FIGURE 2.9: Text to entity identification through TAG.ME

In summary, top down semantic approaches (exogenous techniques) use external knowledge sources to incorporate semantics. Word Sense Disambiguation algorithms process the textual description and replace keywords with semantic concepts (as synsets from WordNet) whereas Entity Linking algorithms focus on the identification of the entities from the text

Table 2.32: Term-Context matrix

	Context								
Items	C1	C2	C3	C4	C5	C6	C7	C8	C9
Beer		✓	✓			✓	✓		
Wine		✓	✓			✓	✓	✓	
Spoon	✓			✓				✓	✓
Glass	✓	✓	✓		✓				✓
Plate	✓	✓	✓		✓				✓

and then linking those entities to other encyclopedic sources such as Wikipedia to create enriched item profiles.

2.2.3.2 Bottom Up Semantic Approaches

These approaches utilize the huge amount of available content (for example, item descriptions, customer reviews) to directly learn a representation of words according to their usage (distribution). The Distributional Hypothesis (Harris, 1954) stated that *"linguistic items with similar distributions have similar meanings"*. This idea of distributional hypothesis led to distributional semantics (FIRTH, 1957) which means that *"by analyzing large corpora of textual data, it is possible to infer information about the usage (meaning) of the terms"*. The semantics learnt according to the usage of a term are therefore called distributional semantics. For example, cat and dog, wine and beer, share a similar meaning since they are often used in similar contexts, as cats are dogs are animals with some similar attributes and beer and wine are both fermented beverages. These methods work on the principle of vector space model (Turney and Pantel, 2010) in which each term is represented. A Term-Context Matrix is created to learn the vector-space representation of each term by encoding the context (for example, a situation) in which the term is used. Consider the matrix in Table 2.32 with items beer, wine, spoon and glass associated with some contexts.

A context can be set according to the granularity required by the representation of term. The granularity can be coarse grained (a whole document) or fine grained (a paragraph, sentence, a window of words). An example of coarse grained context could be, a document of customers' review about the quality of beverages and the cutlery served at the restaurant they dined at. A fine grained context can be the description of items purchased by a customers' over the weekend. In Table 2.32, each row represents a vector. We can observe

Table 2.33: Entity/concept-context matrix

	Context								
Concepts/Entities	C1	C2	C3	C4	C5	C6	C7	C8	C9
H.G. Wells		✓	✓			✓	✓		
Stephen King		✓	✓			✓	✓	✓	
Horror	✓	✓	✓	✓				✓	
Comedy					✓				✓

that beer and wine have a good overlap (they occur in similar contexts), so they are similar, whereas beer and spoon have no overlap, so they are not similar. Based on this, we can say that beer and wine will be more close to each other in the vector space as compared to beer and glass. A vector space representation (called WordSpace) is learnt according to terms' usage in contexts. Specifically, terms sharing a similar usage are very close in the word space. An example of Wordspace (Lowe) for the Term-Context Matrix in Table 2.32 is shown in Fig. 2.10.

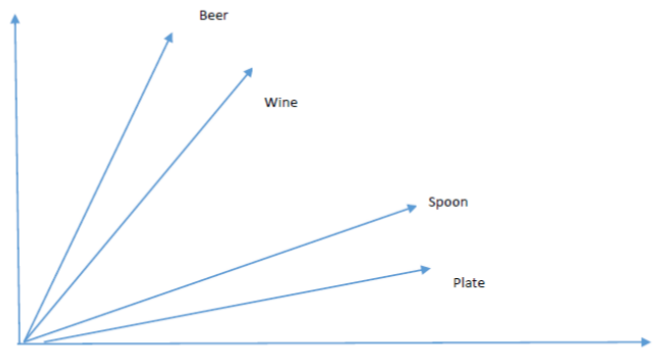


Figure 2.10: Wordspace

Similarly, Distributional semantic models provide the flexibility to replace keywords with concepts as synsets or entities and then semantic similarities can be computed between them. For example, H.G. Wells and Stephen King will be closer to each other as they both share similar interests in writing science fiction, fantasy and suspense novels as shown in Table 2.33.

Given a WordSpace, a vector space representation of documents (called DocSpace) is typically built as the centroid vector of word representations, that is, a document vector will be represented as the average vector of the words in the document. Once we obtain the vector representations, it is possible to perform similarity calculations between items according

Table 2.34: ESA matrix

	Contexts				
Terms	Context1	Context2	Context3	Context4	Context5
Big Data	0	0.26	0	0.01	0.01
Python	0.42	0	0.01	0.01	0
R	0	0.15	0.01	0.02	0.02

to their semantic representation. A common similarity measure is cosine similarity.

In summary, we can utilize the (big) corpora of data to directly learn a semantic vector-space representation of the terms of a language. It facilitates to infer lightweight semantics, which are not formally defined such as those in Wikipedia. Furthermore, they are flexible as items, concepts, entities are represented as vector and similarities can easily be computed between terms or terms and documents. The adoption of context can have different granularities which can span a whole document or shrink to a sentence, paragraph or window of words. However, the limitations are that in order to learn these semantic representations, huge amount of content is needed. The resultant term-context matrices are particularly huge and difficult to build as it may have lots of features which cause the need for dimensionality reduction.

2.2.3.2.1 Explicit Semantic Analysis ESA (Gabrilovich and Markovitch, 2009) is a distributional semantic model which uses Wikipedia articles as context to find the usage of terms. A term-context matrix known as ESA matrix is created where each column represents a context (a Wikipedia article). Each entry in the matrix represents the relatedness of a term to the context computed using the TF-IDF score (details on TF-IDF computation can be found in Sect. 2.2.1.1). Each row of the ESA matrix is called semantic interpretation vector of a term t. As a result, each Wikipedia page can be described in terms of the words with the highest TF-IDF score. Consider Table 2.34, which shows an ESA matrix, where each context represents a Wikipedia article about a particular concept. For example, context 1 represents article about the concept of "Python" programming language. Each entry in the matrix shows association between the term and the concept by using the TF-IDF score.

Each row in the matrix represents the semantics of a word as a vector of its associations with Wikipedia concepts. For example, the second row in the matrix represents the association of the term “Python” across 5 Wikipedia articles. The highest the score, the more the strength of its ‘semantic connection’ with a Wikipedia concept. Similarly, each column represents the semantics of the article as a vector of the terms in the article. A semantic representation of an item (for example, movie, product, document, sequence) can be built as the centroid vector of the semantic interpretation vectors of the terms in the item description (for example, product title or description, movie plot). Next, to find the semantic relatedness (similarity) between a pair of text fragments (for example, description of two products), cosine similarity can be computed between their semantic interpretation vectors. The higher the cosine similarity, means items are related closely.

In summary, ESA models are Distributional Model which uses Wikipedia Article as context. They provide a very transparent representation where columns have an explicit meaning. The downside is that the whole matrix is very huge and requires tuning of the parameters. For example, the choice about the number of articles, number of terms to represent the article etc.

2.2.3.2.2 Random Indexing Dimensionality reduction methods address the huge matrix dimensionality issues. They learn a more compact vector-space representation of terms and items while still maintaining the important information. These techniques also known as word embedding techniques where embedding represents the words in a smaller dimension as compared to their original representations. Random Indexing (Sahlgren, 2006) is an incremental and scalable technique for dimensionality reduction. The algorithm works in four main steps (i) Assign a vector to each context (word, documents, etc.).The length (dimension) of the vector can vary according to the requirement,(ii) fill the vector with (almost) randomly assigned values,(iii) given a word, collect the contexts where that word appears and (iv) sum the context and obtain the final representation of the word. The resulting representation is a smaller but (almost) equivalent to the original one. The steps are explained in Example 2.9.

Example 2.9.Random Indexing

Input: context granularity, such as,(document, paragraph, sentence, word), dimension

Figure 2.11: Matrix for Random Indexing

$$\begin{array}{c} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ \vdots \\ r_n \end{array} \begin{bmatrix} 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Table 2.35: Computation for semantic vector representation of term t1

r1	0	0	-1	1	0	0	0
r2	1	0	0	0	0	0	-1
r5	1	0	0	-1	0	0	0
t1	**2**	**0**	**-1**	**0**	**0**	**0**	**-1**

size =k =8, Possible values, for example, $\{-1, 0, +1\}$

Output: Reduced Matrix with each term represented as a context vector of k-dimension

Step 1: Define the granularity of the context

Consider a sentence as the context.

Step 2 – Create the random matrix R

For example, Table 2.11. shows the matrix where row 1 (r1) represents a context (a sentence in our example) and column represents terms. Each 'context', that is, sentence is assigned random values in the range -1,0,+1 across k dimensions. Each row represents a context vector across 8 dimensions.

Step 3 –Building vector space representation of a term ***t***

It is obtained by combining the random vectors of the context in which it occurs. For example, assume a term t_1 occurs in r_1, r_2 and r_5 , that is $t_1 \in \{_1, r_2, r_5\}$, then the vector representation of the term t1 will be computed by taking the sum of vectors of r_1, r_2 and r_5 as shown in Table 2.35. As an output of this step, we obtain the wordspace, that is, each term (word) is represented in a vector space.

Step 4 – Building the document space

In a similar way, by aggregating the vector space representations of the terms occurring in a document d, we can represent the document d in the vector space. The output of this

Table 2.36: WordSpace and DocSpace

Word Space								DocSpace						
	c1	c2	c3	c4	...	ck			c1	c2	c3	c4	...	ck
t1								d1						
t2								d2						
t3								d3						
t4								d4						
...								...						
tn								dn						

step is the document space where each document is represented as a k-dimension vector which represents the semantic associations between the terms and the documents. Table 2.36 shows a uniform representation of the wordspace and docspace.

In summary, Random Indexing is incremental and scalable technique for learning word embeddings, that is, representing words in a reduced dimension. The advantages are it provides smaller vector space representation and dimension of the space can be set arbitrarily. However, the limitation is that the learnt representations are not transparent anymore, that is, we are unaware what “concepts” are used to find associations between terms and terms and documents. In addition to that, proper parameter tuning is required to find the optimal size of the embeddings.

2.2.3.2.3 Word2Vec Word2vec (Mikolov et al., 2013) is another distributional model for learning semantic representation of words. It is a neural network based model which consists of input, hidden and output layers. Word2vec represent words by a vector of real numbers of dimension d. Training of the model is based on the Skip-Gram methodology (Mikolov et al., 2013) which, when given an input of a word $w(t)$, predict its context words (neighbors) that is, $w(t-2), w(t-1)\ldots w(t+1), w(t+2)$.
The context of a word w within a sentence is the set of x surrounding words. For example, for the sentence “My new iphone fell out of my pocket and broke its screen” we have the target word “iphone”. With a sliding window of size=1, moving along a sentence, the skipgram model (Mikolov et al., 2013) predicts the probabilities of a word being a context word given a target word “iphone”, where context words are those on the left and right of the target word within the sliding window. In this case, with a context, $x = 2$, for the word

"iphone" the context words are "My", "new", "fell", "out" that is, two words each towards the left and right of the target word "iphone".

A context-target pair is considered as a new observation (training sample) in the data. For example, the target word "iphone" in the above case produces four training samples as (my, iphone), (new, iphone), (fell, iphone) and (out, iphone). This process is iterated for each word in the sentence. Word2Vec is a distributional model since it learns a representation such that couples (context, word) appearing together have similar vectors.

The algorithm is fed with a corpus, and training examples are created through Skip-Gram. For example, for the above sentence, training samples will be generated as (new, the),(new, iphone), (iphone, new),(iphone, fell)and so on by using a window size of 1. The model aims at maximizing the probability of predicting a context C given a word w by using Eq. 2.15.

$$\arg\max_{\theta} \sum_{(w.c)\in D} \log p(c \mid w) \tag{2.15}$$

The probability is high when scalar product is close to 1, that is, when vectors are similar. Some of the generated (context, word) pairs are (([my,iphone],new), ([new,fell],iphone), ([iphone,out],fell). The errors are collected and the weights in the network are updated accordingly. Stochastic Gradient Descent or Mini-Batch (every 128 or 512 training examples) are used for model updates. The steps for predicting the probability of a context(s) given a target word using Word2Vec are explained in Example 2.10.

Problem: Given an input of a word $w(t)$, predict its context words (neighbors), $w(t-2), w(t-1), ..w(t+1), w(t+2)$

Example 2.10. Word2Vec (Skipgram Model)

Word2vec uses a neural network with an input, hidden and output layer as shown in Fig. 2.12. The input layer has the same number of neurons as the number of words in the vocabulary (unique words in the sentences). The size of the hidden layer is according to the dimensionality (length of the word vectors). The input and the output layer have the same size. Therefore, if the vocabulary for learning word vectors consists of V words and N is the dimension of word vectors, then the connections from the input to the hidden layer can be represented by matrix W_I of size $V \times N$ where each row representing a vocabulary word. In the same way, the connections from hidden layer to output layer can be described by a matrix W_O of size $N \times V$. Here, each column of W_O matrix represents a word from the given vocabulary.

Given the vocabulary size V, word embedding vectors of size N will be learned. The model learns to predict one context word (output) using one target word (input) at a time.

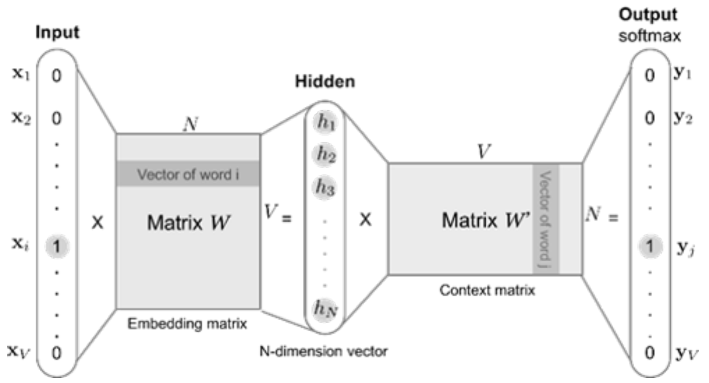


FIGURE 2.12: Word2vec neural model

Consider a set of three sentences “The lion saw a deer”. “The lion chased the deer”. “The deer disappeared in the bush”. Our goal is to predict the context words (neighbors) of the target word “disappeared” when the window size is 1. Below are the steps illustrating the training and learning process of Word2vec.

Input: sentences, V = size of vocabulary (unique items in the data), C = number of context words, N = dimension (length of the vector), Input layer size – [1 X V], Input-hidden weight matrix = Wi =[V x N], Number of neurons in hidden layer – N, Hidden-Output weight matrix size – [N X V], Output layer size – [1 X V]

Output: probability of predicting the context words for each target word. In our running example, we want to predict the context words for the target word “disappeared”.

Step 1: Vocabulary creation

The first step is vocabulary creation which consists of all the unique words in the input sentences. After alphabetically arranging all the unique words in the input sentences, the vocabulary consists of nine words indexed by their position as $\{1 : a,\ \ 2 : bush,\ \ 3 : chased,\ \ 4 : deer,\ \ 5 : disappeared,\ \ 6 : in,\ \ 7 : lion,\ \ 8 : saw,\ \ 9 : the\}$. For the input layer to take the target word as input, it needs to be represented as 1-V (hot vector) representation, which is $[0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0]$, where 1 shows the position of the word in the vocabulary. Similarly, the actual output that we expect from the system which is the context word "deer" has a 1-V vector representation as $[0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0]$. The input vector is represented as X and the output vector is represented as Y.

1.1.Initializing input (W_i) and output weight matrices (W_o)

Initially the weights are randomly chosen. There are two sets of weights, one is between the input and the hidden layer (W_i) and second between hidden and output layer (W_o). Input-Hidden layer weight matrix size = $[V \times N]$, hidden-Output layer weight matrix size = $[N \times V]$: Where N is the number of dimensions to represent the word. It is arbitrary and a hyper-parameter for a Neural Network. Also, N is the number of neurons in the hidden layer. Here, N=3 and V = 9.

$$\mathrm{Wi} = \begin{bmatrix} -0.0945 & -0.4440 & 0.3139 \\ -0.4908 & -0.2299 & 0.0655 \\ 0.0729 & 0.1722 & -0.3578 \\ 0.1045 & -0.4630 & 0.0794 \\ -0.2261 & -0.1547 & -0.0384 \\ 0.4061 & -0.1928 & -0.4420 \\ 0.1818 & 0.0883 & 0.2776 \\ -0.0553 & 0.4918 & 0.2631 \\ 0.1045 & -0.4630 & 0.0794 \end{bmatrix},$$

$$\mathrm{Wo} = \begin{bmatrix} 0.0231 & 0.4799 & 0.4321 & 0.3755 & -0.3647 & -0.1198 & 0.2661 & -0.3510 & 0.3755 \\ -0.3680 & 0.4248 & -0.2571 & -0.1488 & 0.0339 & 0.3539 & -0.1449 & 0.1309 & -0.1488 \\ 0.4224 & 0.3645 & 0.4679 & -0.0203 & -0.4239 & -0.4388 & 0.2685 & -0.4468 & -0.0203 \end{bmatrix}$$

Step 2: Computing the vector at the hidden layer

Now we have the input vector X representing the word *disappeared*, the output at the hidden layer neurons can be computed as:

$$\mathbf{H_t} = \mathbf{X_t} * \mathbf{W_I} = \quad [-0.22608 \quad -0.154659 \quad -0.038422]$$

Step 3: Computing the vector at the output layer Next, the vector for output layer will be computed by taking the product of the hidden vector obtained in step 2 with the output weight matrix (W_o) which will give the result,

$$\mathbf{O}_t = \mathbf{H}_t * \mathbf{W}_2 = \begin{bmatrix} 0.035 & -0.188 & -0.075 & -0.061 & 0.09 & -0.01 & -0.048 & 0.076 & -0.061 \end{bmatrix}$$

This output vector has a combination of positive and negative values. To understand the probability for each word being the context word, these values will be converted to probabilities using the softmax function with the following equation, where x refers to the value we require to be in the range [0-1].

Softmax $= \sigma(x) = \frac{1}{1+e^{-x}} = \sigma(0.035) = \frac{1}{1+e^{-0.035}} = 0.497.$

Similarly, we will compute the sigmoid for other values and the final out put vector will be:

$$\mathbf{O}_t = \begin{bmatrix} 0.497 & 0.453 & 0.481 & \mathbf{0.519} & 0.523 & 0.509 & 0.488 & 0.485 & 0.485 \end{bmatrix}$$

This output vector shows the probability score of each word in the vocabulary, that is, a , bush, chased, deer, disappeared, in, lion, saw, the to be the context word for the target word *disappeared*. So, according to this output, we can see that the model predicts *deer* as the context word to the target word *disappeared* with the highest probability score. In this case, the model has correctly predicted the context word for the target word. This process is repeated for each word in the vocabulary.

$\mathbf{O}_t =$

a	bush	chased	deer	disappeared	in	lion	saw	the
0.497	0.453	0.481	**0.519**	0.523	0.509	0.488	0.485	0.485

In summary, Word2vec learns Word Embeddings through Neural Networks, it is not based on counting co-occurrences rather it relies on predicting the distribution of words. The advantages of this approach are that the representation can be really small (size<100) and the semantic vector representations of words are very accurate. On the other hand, the drawback of these models are that the learnt embeddings are not transparent in terms of providing explanations. Additionally, they needs more computational resources.

2.2.3.2.4 Prod2Vec The prod2vec (Grbovic et al., 2015) model uses purchase sequences to learn vector representation of products and then compute the semantic similarities between products. In particular, prod2vec uses the same principle of Word2Vec by applying skip gram model (Mikolov et al., 2013) to learn product representations and aims to maximize the objective function over the entire set S of purchase sequences, defined in Eq.2.16

$$\mathcal{L} = \sum_{s \in S} \sum_{pi \in s} \sum_{-c \leq j \leq c, j \neq 0} \log \mathbb{P}\left(p_{i+j} \mid p_i\right) \tag{2.16}$$

Where products from the same purchase sequence are in random order. Probability of $\mathbb{P}\left(p_{i+j} \mid p_i\right)$ observing a neighboring product p_{i+j} given the current product p_i is defined using the function as in Eq.2.17

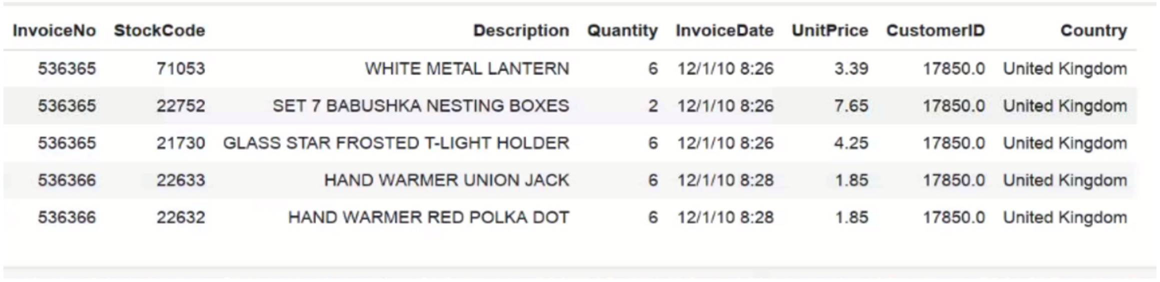
$$\mathbb{P}\left(p_{i+j} \mid p_i\right) = \frac{\exp\left(v_{pi}^{T} v_{i+j}'\right)}{\sum_{p=1}^{p} \exp\left(v_{pi}^{T} v_{p}^{J}\right)} \tag{2.17}$$

Where v_p and v_p' are the input and output vector representations of product p, c is the length of the context for product sequences, and P is the number of unique products in the vocabulary created from purchased sequences. Prod2vec models the context of purchase sequences, where products with similar contexts that is, with similar neighboring purchases will have similar vector representations and are closer to each other. According to the analogy of Word2vec each a purchase sequence is a sentence and product id's are words, that will be fed to the model for training. The steps to learn product co-occurrences in a context given a target product using Prod2Vec are explained in Example 2.11.

Example 2.11. Learning Products' Vector Representations through Prod2Vec
Step 1: Creating purchase sequences

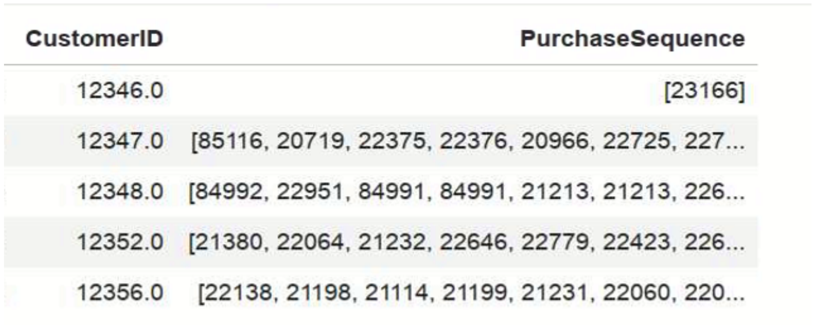
Purchase sequences are created according to time stamp, grouped by customers (Fig.2.14) from the Online Retail Data set that shows transactions by customers (Fig. 2.13). In the data set, a tuple is of the form (InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country). Data was preprocessed for duplicates, null values and only relevant columns (StockCode, InvoiceDate, CustomerID) were kept for further steps.

The idea is to have a given product (target product) and finding products which are in its neighborhood (context). This will be achieved by learning embeddings which is dense vector representation of items.



InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/10 8:26	7.65	17850.0	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/10 8:26	4.25	17850.0	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	12/1/10 8:28	1.85	17850.0	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/10 8:28	1.85	17850.0	United Kingdom

FIGURE 2.13: Online Retail data set sample records



CustomerID	PurchaseSequence
12346.0	[23166]
12347.0	[85116, 20719, 22375, 22376, 20966, 22725, 227...
12348.0	[84992, 22951, 84991, 84991, 21213, 21213, 226...
12352.0	[21380, 22064, 21232, 22646, 22779, 22423, 226...
12356.0	[22138, 21198, 21114, 21199, 21231, 22060, 220...

FIGURE 2.14: Online Retail data set after preprocessing

Step 2.1.Learning Embeddings and model training

Consider a purchase sequence S_1 with products $P_1, P_2, P_3, \ldots, P_n$ sorted according to the timestamp. Each product will be represented as a feature vector across 4 dimensions learnt from prod2vec model. Each sequence will be represented as aggregated feature vector based on all the products in the purchase sequence. Below steps show the detailed steps for learning vector representation for products.

Input: sentences (purchase sequences from Fig. 2.14, context C =2, N = 4, Input layer size – $[1 \times V]$, Input hidden weight matrix size – $[V \times N]$, Number of neurons in hidden layer – N, Hidden-Output weight matrix size – $[N \times V]$, Output layer size – $C[1 \times V]$, V= size of vocabulary (unique items in the data), C = number of context products.

Output: probability of each unique product to be in the neighborhood of the context product. The created sentences (purchased sequences) will be fed to the model for learning product vector representations.

2.2. Vocabulary creation

From the purchase sequences, vocabulary of all unique products will be created based on the product id's. For example, here we consider a small vocabulary size (V = 10) with products created from the purchase sequences as 1: ‘85116’ , 2: ‘22951’, 3: ‘23166’, 4: ‘22376’, 5: ‘21380’, 6: ‘21232’, 7: ‘22423’, 8: ‘22060’, 9: ‘22646’, 10: ’22632’

2.3. Input layer weight matrix

Initially the weights are randomly chosen. The input layer and the target, both are one-hot encoded of size $[1 \times V]$. Here $V = 10$. Model takes input (product id) and represent it in the form of one hot vector encoding fashion, that is, according to its position in the vocabulary. For example one hot vector representation of product with id ’22951’ will be [0100000000] as it is at second position in the vocabulary. The dimension of the vector will be according to the size of the vocabulary which is 10 in this case.

There are two sets of weights, one is between the input and the hidden layer and second between hidden and output layer. Input-Hidden layer weight matrix size = $[V \times N]$, hidden-Output layer weight matrix size = $[N \times V]$: Where N is the number of dimensions to represent the product. It is arbitrary and a hyper-parameter for a Neural Network. Also, N is the number of neurons in the hidden layer. Here, N=4 and $V = 10$. Weight matrices are shown as U (input to hidden layer) and W (hidden to output layer) respectively.

$$U = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 \\ 33 & 34 & 35 & 36 \\ 37 & 38 & 39 & 40 \end{bmatrix} W = \begin{bmatrix} 0.12 & 0.13 & 0.14 & 0.15 & 0.16 & 0.17 & 0.18 & 0.19 & 0.2 & 0.21 \\ 0.22 & 0.23 & 0.24 & 0.25 & 0.26 & 0.27 & 0.28 & 0.29 & 0.30 & 0.31 \\ 0.32 & 0.33 & 0.34 & 0.35 & 0.36 & 0.37 & 0.38 & 0.39 & 0.40 & 0.41 \\ 0.42 & 0.43 & 0.44 & 0.45 & 0.46 & 0.47 & 0.48 & 0.49 & 0.50 & 0.51 \end{bmatrix}$$

The input is multiplied by the input-hidden weight matrix U and called hidden activation. It is simply the corresponding row in the input-hidden matrix which is 2nd row in our case. So multiplying, [0101000000] with the input hidden weight matrix U, we get the hidden input vector (H_t) as: $\mathrm{H_t} = \begin{bmatrix} 5 & 6 & 7 & 8 \end{bmatrix}$

Next step is to multiply the hidden input computed (H_t) with the hidden- output weight matrix W to obtain the output ($\hat{O}_t$).
Multiplying hidden input [5 6 7 8] with hidden output weight matrix W, we get the output vector as:

$$\hat{O}_t = \begin{bmatrix} 7.52 & 7.78 & 8.04 & 8.3 & 8.56 & 8.82 & 9.08 & 9.34 & 9.6 & 9.86 \\ 7.52 & 7.78 & 8.04 & 8.04 & 8.04 & 8.82 & 9.08 & 9.08 & 9.6 & 9.86 \end{bmatrix}$$

Now converting the scores obtained in the output vector $\hat{O}_t$ into probabilities by using $Softmax = (x) = \frac{1}{1+e^{-x}}$ as the sum of probabilities of all unique items has to be in the range 0 and 1. The final obtained probabilities for each of the products (in the vocabulary) being the context of the target product are shown below.

$$O_t = \begin{bmatrix} 0.024 & 0.030 & 0.040 & 0.051 & 0.067 & 0.087 & 0.1133 & 0.1470 & 0.1906 & \mathbf{0.2473} \end{bmatrix}$$

So, here our model predicted product with id "22632" to be the context(neighborhood) of the target product "22951", which is a wrong prediction, as the correct context products (actual) are products "85116" and "23166" with their one-hot vectors are represented as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -0.98 & 0.030 & 0.040 & 0.051 & 0.067 & 0.087 & 0.1133 & 0.1470 & 0.1906 & 0.2473 \end{bmatrix}$$

$$\begin{bmatrix} 0.024 & 0.030 & -0.95 & 0.051 & 0.067 & 0.087 & 0.1133 & 0.1470 & 0.1906 & 0.2473 \end{bmatrix}$$

Element-wise sum is taken over all the error vectors to obtain a final error vector as :

$$\begin{bmatrix} -0.95 & 0.06 & -0.919 & 0.103 & 0.134 & 0.174 & 0.226 & 0.294 & 0.38 & 0.494 \end{bmatrix}$$

This error vector is propagated back to update the weight matrix. The weight between the hidden layer and the output layer is taken as the vector representation of the product across N dimensions. Final vector representations for all ten products as shown in Product Vector PV matrix, where each column represents vector representation of a product p across N (4) dimensions.

$$PV = \begin{bmatrix} 0.24 & 0.13 & 0.14 & 0.15 & 0.21 & 0.17 & 0.18 & 0.19 & 0.2 & 0.21 \\ 0.22 & 0.23 & 0.15 & 0.10 & 0.26 & 0.27 & 0.28 & 0.29 & 0.30 & 0.31 \\ 0.32 & 0.13 & 0.24 & 0.15 & 0.36 & 0.37 & 0.38 & 0.35 & 0.40 & 0.41 \\ 0.12 & 0.23 & 0.14 & 0.25 & 0.46 & 0.47 & 0.48 & 0.49 & 0.50 & 0.51 \end{bmatrix}$$

Here, column 1 is vector representation of product “85116”, column 2 represents vector representation of product “22951” and so on.

Step 3: Product Recommendation

The model is trained and has learnt the product vector representations. So now for any given (target) product, it will look up its vector representation and recommend the product whose vector representation is close to it by calculating the cosine similarity between their vectors.

In summary, Prod2Vec utilizes the same concept of Word2Vec and applies skipgram method (Mikolov et al., 2013) on purchase sequences to learn the product semantics by predicting the probability of context(s) (neighboring products) given a target product. Once, the vector representations are learned, similarities can be computed to find closeness between products. This approach considers the semantic similarity between products, however it has some limitations such that, it do not consider the sequential relationship and the temporal information between the products. Also, it do not include the purchase history of customer to cover both long term and short term preferences. Further more, it does not incorporate the knowledge transfer from different customer purchase histories in order to determine similar buying habits among customers.

Chapter 3

Problems Identified and Research Questions

3.1 Problems

From the discussions in Chapter 2, following problems and issues have been identified.

1. Recommendation Systems suffer from the following problems:

    (a) The **Cold Start** problem- In cold start problem items cannot be recommended due to the lack of information about the user and/or the item. Cold-start problem presents a collective issue of new item and new user to RS. There are two types of cold start (i) New Item: A new item cannot be recommended initially when it is introduced to a Collaborative Filtering system with no ratings. For instance, MovieLens (movielens.org) cannot recommend new movies until these have got some initial ratings. (ii) New User: When a new user enters the system, it is bit hard to find similar users or to create a content based profile as previous preferences (such as browsing history, likes/dislikes) of user are not available.

    (b) **Sparsity** – Sparsity occurs when majority of the users do not rate most of the items and consequently the ratings matrix becomes very sparse which limits the chances of finding a set of users with similar ratings. In other words, this problem arises when there are many items to be recommended but only few recommendations are provided (because of fewer available ratings on items).

   (c) **Content Overspecialization and Diversity** - This means that the system cannot provide something unexpected. Such systems would not find anything novel since they only recommend items similar to the user profile. This problem affects the user and the system. The user may get bored easily and not return. The system may not be able to introduce new items. Differently speaking, generally, a user can opt for an item of his interest from a recommendation list if the list reflects some diversity in the recommended items to some extent. Seamless recommendations for a restricted type of product have no value until or unless it is desired or explicitly described by the user with a narrow set of preferences. In the initial stage, when the RS is used as a knowledge discovery tool, the users may wish to explore new and different options available. So, it is required to have such solution that could achieve the goal of diversity of items together with the accuracy of recommendation.

2. **Sequential Pattern** based recommendation methods have the following limitations:

   (a) **Huge Search Space-** SPM algorithms have a huge search space when applied on customer transactions. A method is required to efficiently guide the algorithm in this search space.

   (b) **Lack of Contextual Information -** Sequential rules generated from frequent sequential patterns lack contextual information, that is, they recommend items by finding a match between the target product and the generated rule. If a match exists, only then a product is recommended. Sequential rules generated from sequential patterns do not consider relationships between items according to their context. For example, if there are two sequential rules for chocolate purchase such as: Ferrero Rocher, Ferrero Rondnoir $\rightarrow$ Rafeallo and (b) Kinder Chocolate Bar, Kinder Surprise $\rightarrow$ Kinder Beuno, where rule (a) (indicating that a purchase of Ferrero Rocher, Ferrero Rondnoir will most likely lead to the purchase of Rafeallo), so for a new purchase sequence where a customer bought Nutella and Rafeallo, he can be recommended to purchase "Ferrero Collection" as they all belong to same brand and have "similar" characteristics such as their ingredients, however if there is no rule for the product “Nutella”, the customer will not be recommended with a product. Conventional sequential rules fail to capture such semantic relationships between products and will only recommend product that exist as a result of a matching rule with products “Nutella and Rafeallo”. Therefore, it cannot recommend products that are similar in semantics as they are based only on the frequent sequential occurrence. i.e., frequency count based on a minimum support threshold.

So, there is a need to learn inherent meanings of items to find similarities between products which will have an influence on succeeding purchases and diverse product recommendation.

(c) **Lack of personalization** – Sequential patterns are not targeted for a specific customer, as they infer decisions based on a global view of sequences extracted from the users’ historical data.

3.2 Research Questions

In light of the problems identified, following research questions are formulated which will be answered by this dissertation.

Research Question 1: How incorporating data from multiple sources can contribute towards personalized recommendations?

Answer: This thesis proposes a comprehensive model which inputs customers’ purchase history and products’ meta data (e.g., title, description and brand) and extract products’ sequential and semantic knowledge according to their (i) usage (e.g., products co-purchased or co-reviewed) and (ii) textual features by finding similarity between products based on their characteristics using distributional hypothesis methods (Prod2Vec, Glove, Doc2vec and TF-IDF) which consider the context of items’ usage. Products’ semantic feature vectors were learnt by training various distributional models and obtaining products’ representations by: (a) Individually training Prod2vec (Grbovic et al., 2015) and Glove (Pennington et al., 2014) models using products from customers’ purchase sequences sorted according to the time stamp, (b) utilizing product feature vectors (embeddings) obtained from prod2vec (Grbovic et al., 2015) and Glove (Pennington et al., 2014) to create hybrid embeddings for product representation, (c) individually training TF-IDF (Salton, 1988) and Doc2Vec (Mikolov et al., 2013) models using more product features such as title, description and brand from products' meta data and customers' purchase sequences, where a document represents the collection of products' title, description and brand purchased by the customer sorted according to the time stamp, (d) utilizing product features (embeddings) obtained from TF-IDF (Salton, 1988) and Doc2Vec (Mikolov et al., 2013) to create hybrid embeddings for product representation (Sect. 5.1, Algorithm 2).

The extracted products' semantic knowledge is then utilized in the candidate generation and recommendation phases to generate candidate items that are similar in semantics with target user's preference to generate recommendations that are of interest to user. The items' semantic knowledge is integrated in the candidate generation and recommendation process of sequential recommendation to (i) mine frequent sequential purchase patterns (Sect. 6.1, Algorithm 3), (i) enrich item to item similarity matrix of Collaborative Filtering method and (ii) enhance sequential next item prediction process of first order Markov Model by incorporating the semantic knowledge (semantic similarity) into the Transition Probability Matrix (Sect. 6.1, Algorithm 4) which provides recommendations tailored towards customers' interest.

Research Question 2: Can items' semantic knowledge be integrated to enhance the performance of item based Collaborative Filtering Systems?

Answer: Yes. This is obtained by computing cosine similarity between the products' semantic feature vectors obtained from customers' purchase histories and products' meta data. This semantic similarity between items is then used to create an item-item semantic similarity matrix for item based Collaborative Filtering (Chapter 6). To further enhance the item based Collaborative Filtering with items semantic and sequential relationships, a weighted products' score measure is proposed which is computed through semantic similarity, confidence and lift measures between a pair of products. The item similarity matrix is then populated with this score measure where each entry in the matrix represents semantic and sequential relationship (in terms of a score value) between a pair of products (Sect. 7.1, Algorithm 6). The matrix can then be used to generate semantic and sequential next item recommendation by taking each item in target user's profile and retrieving a list of Top-K items with a high score.

Research Question 3: How does incorporating semantics to sequential pattern mining methods address their limitations and enhance the accuracy of recommendations?

Answer: This is accomplished by integrating semantic knowledge extracted from customers' purchase histories and products' meta data to enhance the mining process. The first step is to determine Top-N customers based on semantic similarities between products in their purchase sequences, where each purchase sequence will be represented as an aggregate vector of all products in the sequence (Section 6.1.3). This is analogous to representing a sentence as collection of words and then creating a database of these semantically similar purchase sequences to mine frequent semantic and sequential patterns using PrefixSpan (Jian Pei et al., 2001). During the mining process, the semantic information of products from the item similarity matrix along with their support count is used to prune patterns of products that are below the specified semantic similarity threshold and minimum support threshold (Section 6.1, Algorithm 3). Later, in the recommendation phase, the extracted semantic rich sequential patterns are used to generate rules to recommend items that are similar in semantics and can be purchased in sequential order (Algorithm 5).

Research Question 4: Can the inclusion of semantic information solve the problems in the recommendation systems and how?

Answer: To address the items' cold start issue, this thesis proposes to compute semantic and sequential relationships between items from customers' purchase history and products' meta data (e.g., title, description and brand) and extract products' semantic knowledge according to their (i) usage (e.g., products co-purchased or co-reviewed) and (ii) textual features by finding similarity between products based on their characteristics. These relationships obtained in the form of product feature vector representations can then be used to compute similarities between existing and a new item to guarantee the appearance of this newly introduced item into the recommendation set without the use of any explicit ratings.

For a new user issue, where no previous history exists, the current user behaviour (e.g., what items she is browsing), are considered to find items that are similar in semantics.

Then, for recommendation using user-based CF, Top-N neighbors (customers' whose interest is similar to the target customer) are identified based on the products' in their purchase sequence. This is based on semantic similarities between products in their purchase sequences, where each purchase sequence will be represented as an aggregate vector of all products in the sequence (Section 6.1.3) and each user will be represented as an aggregate vector of her purchase sequences to compute Top-N neighbors. This is analogous to representing a sentence as collection of words. *Sparsity* (lack of user item interactions) is addressed as an item's relationship with other items is computed on the basis of item's textual features and usage context (e.g., items co-purchased, co-viewed). Hence, to compute similarities between items, no explicit information (ratings) are required. The huge search space issue is also addressed by narrowing down the search space during the mining process by proposing to prune patterns on the basis of semantic similarity threshold in addition to traditional pruning method of using the support count.

Furthermore, a first order Markov Model's Transition Probability Matrix is enhanced with items' semantic knowledge by first (a) creating a product frequency matrix based on the sequential occurrence of each consecutive pair of products using the customers' historical purchases, (b) creating a Transition probability Matrix by normalizing the matrix obtained in step (a) as explained in Sections 6.2.1 and 6.2.2 and then integrating the semantic information of products from the item similarity matrix by computing a weighted score (based on their semantic and sequential occurrence) for each consecutive pair of products in the Transition Probability Matrix (Section 6.2, Algorithm 4). This provides us the items that are similar in semantics and purchased in sequential order. A personalization vector is then computed for each customer based on any product in its most recent purchases (as a starting probability vector for the customer to begin at state k) in order to determine the probabilities for landing at state k+1 (i.e., recommending a personalized next item by initiating the customers' journey from state k which represent any of the products purchased by the customer during the last time stamp) and then determine the next product for purchase (state k+1) based on the weighted score (Sect. 7.2, Algorithm 7).

Chapter 4

Proposed Semantic Embedded Sequential Recommender for E-commerce Products (SEMSRec)

4.1 Problem Formulation

Consider a set of users $U = \{u_1, u_2, \ldots\ldots, u_n\}$ and a set of products $P = \{p_1, p_2, \ldots\ldots, p_n\}$. Each user u is associated with a sequence of interactions with some items from P such that $S = \{S_1, S_2, \ldots.., S_n\}$. Given a sequence of user-item interactions (click, view, add to cart, purchase), the task of a Sequential Recommendation System is to generate a personalized Top-K ranked candidate item list. The system aims to maximize users' future needs, by considering their long and short term preferences in terms of finding sequential relationships between various user-item interactions (e.g., clicks, views or purchases). In other words, the sequential recommendation task is to learn a complex function f for accurately predicting the probability that user u will choose each product p at time $t + 1$ based on the input behavior sequence and the user profile. Mathematically,

$$O = argmax\ f(I)$$

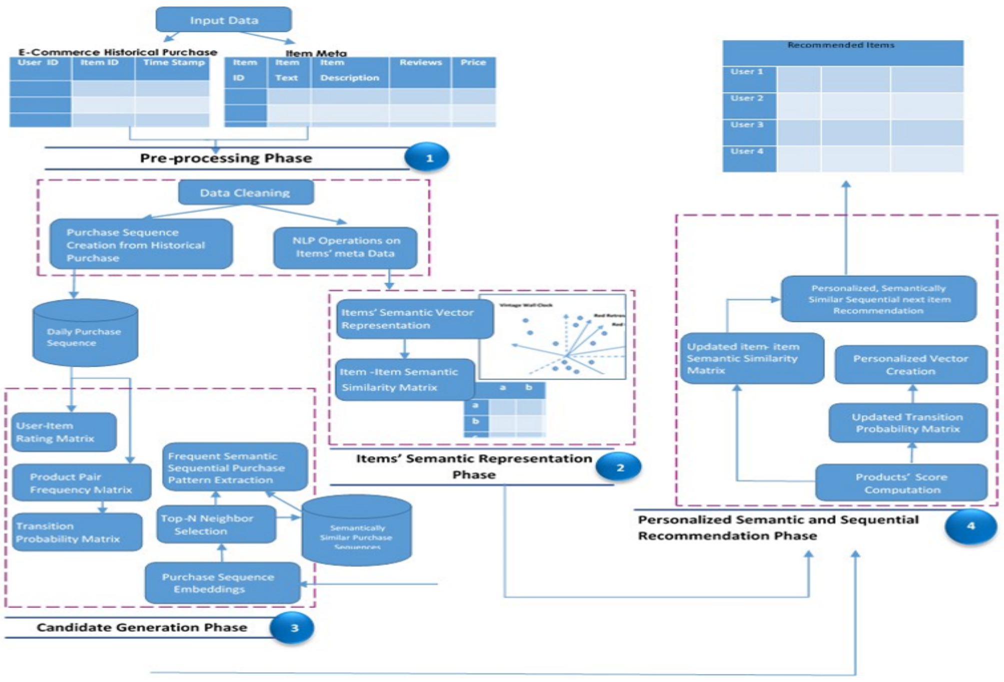
where f represents a utility function which outputs a ranking score for the candidate items, and can be in the form of conditional probability or an interaction score. I is the input sequence such that $I = \{a_1, a_2, ..., a_{|S|}\}$ and represents a sequence of user-item interactions where each interaction $a_i = \langle u, i, p \rangle$ is a triplet comprising of a user u, the user's interaction i, (click, view or purchase) and the corresponding item p. Sometimes, there is some meta data associated with the users (e.g., user demographics or features), items (e.g., product descriptions, title, brand) and user-item interactions (e.g., click, add to the cart, purchase) which can occur under different scenarios (contexts) such as a particular time, location and weather. The output O (i.e., recommendation) is a list of items ordered by the ranking score.

Hence, this work focuses on the task of Top-K recommendations formally stated as, consider a set of users $U = \{u_1, u_2, \ldots\ldots, u_n\}$ and a set of products $P = \{p_1, p_2, \ldots, p_n\}$. Each user u is associated with a sequence of some items from P such that $S_u = (S_1, S_2, \ldots\ldots, S_n)$. Given all users' purchase sequences S_u, the system aims to recommend each user u a list of Top–K tailored products from the candidate set based on the semantic relationship between products instead of $u's$ interactions (ratings) on products. The system aims to maximize users' future needs, by considering their long and short term preferences in terms of finding (a) semantic similarities and sequential relationship between products from customers' purchase histories and product meta data and then (b) integrating products' semantic similarities and sequential relationships into Sequential Pattern Mining, Collaborative Filtering and Transition Probability Matrix of Markov Model to generate semantic and sequential personalized recommendations for customers.

4.2 Proposed Solution and System Architecture

We propose a component based architecture for our proposed system. Fig. 4.1 presents the diagram depicting the architecture of the system and Algorithm 1 shows the processing. The system consists of four main components/phases which are (i) data pre-processing phase, (ii) items' semantic representation phase, (iii) candidate generation (users', items') phase and (iv) semantic and sequential recommendation phase. Here, we provide a brief overview of each component and then in the subsequent chapters (Chapters 5, 6 and 7) will explain the detail functionality along with walk through examples for each component of the proposed system.

The major goal of the proposed Semantic Embedded Sequential Recommendation System (SEMSRec) is to integrate items' semantic and sequential information extracted from customers' purchase histories to compute item similarities for personalized recommendations without using item ratings. It also integrates semantic knowledge to (i) mine semantically rich frequent sequential purchase patterns, (ii) enhance the CF item-to-item similarity matrix and (iii) enrich the transition probability matrix of Markov Model for semantic and sequential next item recommendation(s). Thus, SEMSRec (Algorithm 1) takes customer historic purchase records (HPR), products meta data (title, description, brand), number of Top-N users with similar purchase behavior (N), number of Top-K recommendation items (K), minimum support(s),minimum semantic sim (m_sem), number of steps (L) as input to output set of items that are similar in semantics and purchased frequently in sequential order.



Input Data
E-Commerce Historical Purchase
User ID
Item ID
Time Stamp
Item Meta
Item ID
Item Text
Item Description
Reviews
Price
Pre-processing Phase
1
Data Cleaning
Purchase Sequence Creation from Historical Purchase
NLP Operations on Items' meta Data
Daily Purchase Sequence
Items' Semantic Vector Representation
Vintage Wall Clock
Item -Item Semantic Similarity Matrix
a
b
Items' Semantic Representation Phase
2
User-Item Rating Matrix
Product Pair Frequency Matrix
Transition Probability Matrix
Frequent Semantic Sequential Purchase Pattern Extraction
Top-N Neighbor Selection
Semantically Similar Purchase Sequences
Purchase Sequence Embeddings
Candidate Generation Phase
3
Recommended Items
User 1
User 2
User 3
User 4
Personalized, Semantically Similar Sequential next item Recommendation
Updated item- item Semantic Similarity Matrix
Personalized Vector Creation
Updated Transition Probability Matrix
Products' Score Computation
Personalized Semantic and Sequential Recommendation Phase
4

FIGURE 4.1: Architecture of the proposed system

Algorithm 1 Semantic Enhanced Sequential Recommendation for E-Commerce Products (SEMSRec)

```
Input        : Customer Historic Purchase Records (HPR), Products Meta Data (title, description, brand),number of Top-N users with similar purchase behavior (N), number of Top-K recommendation items (K), minimum support(s),minimum semantic sim (m_sem),number of steps ($L$)
Output       : set of Top-K recommended items (RS)
Intermediates: Customer Purchase Sequences (PS), Products' semantic feature vector representation matrix (PV) of size features $\times$ products,Semantic rich sequential purchase patterns,item semantic similarity matrix $M$, purchase sequence vector ($\overrightarrow{PS_u}$) of user $u$, similar purchase sequence data base (SPSDB),Transition Probability Matrix $P$, Product Frequency matrix $PF$, Semantic rich Transaction Probability matrix $P_1$,Probability Vector for Next Item Recommendation at time t+1 $P^{t+1}$
/* Data Pre-processing */
1  Creating purchase sequences from customers' historical records and cleaning meta data
       using NLP operations (Section 4.2.1 /* Learn Products Semantic Representation */
2  ) Learn products' semantic representation using Algorithm 2 (SKE) in Section 5
/* Generating Semantic rich Sequential Candidate Items */
3  if using Semantic rich Sequential Pattern Mining Method then
4      use Algorithm 3 (SKI_SSPM) in Section 6.1
5  else
6      if using Markov Model Method then
7          use Algorithm 4 (SKI_MM) in Section 6.1.4
/* Generate Semantic and Sequential rich Next Item Recommendations */
8  if using Semantic rich Sequential Pattern Mining Method then
9      use Algorithm 5 (SSR_SPM) in Section 7.1
10 else
11     if using Collaborative Filtering Method then
12         use Algorithm 6 (SSR_CF) in Section 7.1
13     else
14         use Markov Model Method by using Algorithm 7 (SSR_MM) in Section 7.2
```

4.2.1 Data Pre-processing Phase

This component is responsible to pre-process the users' (customers') historical purchase data and products' meta data (title, description and brands) for input to the system. The customers' historical data depends on the purchases made by the customers' over a period of time. The historical data needs to be cleaned and transformed to obtain it in the correct form. The transformations involve filling missing values, removing duplicate records, sorting and grouping customers' purchase sequences according to the timestamp. Other pre-processing task for the product meta data involve natural language processing operations such as a) tokenization (the process of segmenting text into words, clauses or sentences such as separating words and removing punctuations), b) stop words removal (removal of commonly used words unlikely to be useful for learning such as a, the, of), and c) stemming which involves reducing related words to a common stem such as reducing the words loved, loving and lovely to the word love obtained from customer's review about a product expressing her likeliness towards the product. These pre-processing operations need to be performed before the data can be input to the models for learning products semantic representations. This is where the feature extraction, construction, and selection takes place too to get the data model.

4.2.2 Items (Products') Semantic Representation Learning Phase

This phase involves learning the product representations which are then used in the later phases for computing product similarities, semantically similar purchase sequences, extracting semantically frequent sequential patterns and then incorporating this semantic and sequential information into the item-item similarity matrix in Collaborative Filtering for generating Top-K recommendations. The semantics are learned based on:
(a) Product ID's: In this setting, the corpus consists of product id's from customers' purchase sequences sorted according to the time stamp and is used to individually train the Prod2vec (Grbovic et al., 2015) and GloVe (Pennington et al., 2014) models to obtain products' vector representations. Here, for the model training, a corpus of sentences in case of Prod2Vec (where a sentence represents sequence of products purchased by customers sorted according the time stamp) and documents in case of GloVe (where a document represents collection of sequences representing products purchased by the customer) are used.

In a different setting, vectors obtained after training both models are combined (averaged) to obtain a unified feature vector representation of each product in the corpus.
(b) Product's Meta Data: To explore the impact of obtaining product semantics from other product features (textual data), a corpus of documents and tokens in case of TF-IDF (where a document represents collection of products' title, description, brand and tokens

comprise of unique words present in the textual data) and documents for Doc2vec (Mikolov et al., 2013) are used (where a document represents collection of product descriptions, title and brand in a list of list format and each list element represents description, title and brand of a product purchased, and a document ID for each document).

In a different setting, vectors obtained after training both models are combined (averaged) to obtain a unified feature vector representation of each product in the corpus. (Grbovic et al., 2015) is based on word2vec which is a highly scalable predictive model for learning word embeddings from text. It is based on the Distributional Hypothesis, which states that words that appear in the same contexts are close to each other in meanings. A similar hypothesis can be applied in larger contexts such as online shopping where we can treat products as word tokens and use user sequences (analogical to sentences) to learn product embeddings. Word2Vec encode semantics of the words which is exactly what we need for similar products, therefore prod2vec (Grbovic et al., 2015) is used to generate product embeddings. However, the product embeddings generated by prod2vec (Grbovic et al., 2015) only take into account the information of the user purchase sequence, that is, only the local co-occurrence information. Glove model (Pennington et al., 2014) on the other hand is a word vector representation method where training is performed on aggregated global word-word co-occurrence statistics from the corpus (set of all purchase sequences). Therefore to capture information from the purchase sequences at the local and the global level, product embeddings were obtained by training both models (Prod2vec and Glove) individually and then unifying the learnt embeddings for better representation of product vectors. Experiments showed that unified embeddings gave better results in terms of finding similar products.

We further enhanced the products' representations by including other types of item information which is the items' meta data (e.g., product titles, descriptions and brand) and used Doc2vec (Mikolov et al., 2013) and TF-IDF because Prod2vec (Grbovic et al., 2015) and Glove (Pennington et al., 2014) do not take into account these types of information such as the items' metadata. In Doc2Vec model (Mikolov et al., 2013), the words (products) and the paragraph (products' meta data in a purchase sequence) are trained jointly to learn product vector representations (embeddings).

The rationale to obtain product embeddings through aggregating two models (e.g., prod2vec and Glove) on product sequences was to capture information about the products (embeddings) from the purchase sequences at the local and the global level.

Chapter 5 of this thesis explains the step by step detailed functionality of this phase along with walk through example.

4.2.3 Candidate (Users' and Items') Generation Phase

This phase involves generating potential candidates for recommendation and comprises of three steps which are (i) computing products' semantic similarity, (ii) extracting Top-N semantically similar neighbors (users, items) and finally (iii) generating candidates either by (a) mining semantic embedded sequential patterns and rules or (b) using Markov Model created from the daily purchase sequences of customers' historical purchase which is based on proposed semantic transition probability matrix.

Chapter 6 of this thesis provides step by step detailed functionality of this phase for generating potential candidate items and users using both approaches (semantic based sequential pattern mining and semantic Markov Model transition matrix) along with walk through example.

4.2.4 Semantic and Sequential Next Item Recommendation Phase

This phase involves incorporating the semantic and sequential associations between products into the item-item similarity matrix in order to enrich the Collaborative Filtering item-item matrix for recommending products to customers which are similar in semantics and are purchased in sequential order as well. The steps involved in this phase are (i) score computation for products, (ii) creating semantic and sequentially rich (a) item to item similarity matrix in Collaborative Filtering Model and (b) transition probability matrix in Markov Model, (iii) personalized vector creation for target customer and (iv) generating semantically rich and sequential Top-K recommendations.

Chapter 7 of this thesis explains the step by step detailed functionality of this phase to generate semantic and sequential next item recommendations using

semantic based (a) Sequential Pattern Mining, (b) Collaborative Filtering and (c) Markov Model along with walk through examples.

Chapter 5

Semantic Representation Learning of E-commerce Products

In this Chapter, we present the details of obtaining product representations based on product Id's and the meta data using the Prod2vec (Grbovic et al., 2015), Glove (Pennington et al., 2014), Doc2vec (Le and Mikolov) and TF-IDF (Salton, 1988). To explain the input format and the working of the models Prod2Vec(Grbovic et al., 2015), Glove(Pennington et al., 2014), TF-IDF(Salton, 1988) and Doc2Vec(Le and Mikolov), we will use data from Table 5.1 and Table 5.2 representing sample historical product purchase records of customers and products' meta data. For each of these models, we will obtain a semantic representation of items across d = 100 dimensions, where an item can represent documents comprising of products, articles, books, customer reviews or product descriptions. Algorithm 2 shows the steps to obtain semantic representation of products.

5.1 Walk through Examples for learning Products Semantic Feature Vector Representation

Example 5.1. Product's Semantic Representation Using Prod2Vec

Step 1: Creating purchase sequences

From Table 5.1 we create purchase sequences (Table 5.3) for each customer sorted according

Table 5.1: Sample historical product purchase records

Invoice No.	Stock Code	Invoice Date	Customer ID
536365	20674	12/1/10 8:26	17850.0
536365	21242	12/1/10 8:26	17850.0
536365	20675	12/1/10 8:26	17850.0
536365	21245	12/1/10 8:28	17850.0
536365	20677	12/1/10 8:28	17850.0
536365	20655	12/1/10 8:30	17850.0
536365	20677	12/1/10 8:30	17850.0

Table 5.2: Sample of product meta data

Stock Code	Title	Description	Brand
20674	Green polka Dot bowl	Earthenware, largest measures5.5 inch h x 12 inch l x 11.25 inch hand wash	Tag limited
21242	Red retrospot Plate	These beautiful plates are composed of high-rated heavyweight plastic materials rendering the plates leak-free, soak resistant, cut proof and unbreakable.	Silver Spoons
20675	Blue Polka Dot bowl	This polka dot bowl is fun and festive and perfect for that bowl of cereal in the morning or bowl of ice cream in the evening. It is finished in a blue celadon glaze with a sprinkling of matte black polkadots. Dish washer safe	Creative innovations
21245	Green polka Dot plate	Add a splash of color with this bright party detail! Green and White Dots Dessert Plates (8), 7"	Party2u
20677	Pink polka Dot bowl	Earthenware,largest measures 5.5 inch h x 12 inch l x 11.25 inch. Hand wash	Tag limited
20655	Queen of skies Luggage Tag	Suitcase tag made PU material, in front with a protective film. waterproof. Fully Bendable / Flexible Material to Prevent Breaking or Losing Your leather luggage tags.	PAGSRAH

Algorithm 2 Products' Semantic Knowledge Extraction (SKE)

```
Input  : Customer Historic Purchase Records (HPR), Products Meta Data (title, description, brand)
Output: Customer Purchase Sequences (PS), Products' semantic feature vector representation matrix (PV) of size features $\times$ products
1 Create purchase sequences from Customers' historical purchase records
2 Obtain Matrix ($PV_{prod2vec}$) of features $\times$ items $\leftarrow$ computed using Prod2vec (Example 5.1)
3 Obtain Matrix ($PV_{glove}$) of features x items $\leftarrow$ computed using Glove (Example 5.2)
4 Obtain Matrix ($PV_{tfidf}$) of features $\times$ items $\leftarrow$ computed using TF-IDF (Example 5.3)
5 Obtain Matrix ($PV_{Doc2Vec}$) of features $\times$ items $\leftarrow$ computed using Doc2vec (Example 5.4)
6 Create hybrid PV matrix ($PV_{hybrid}$) $\leftarrow$ computed by averaging the embeddings (semantic
    feature vectors) of each product in their respective PV matrices
```

Table 5.3: A purchase sequence database (Purchase Sequences)

SID	Purchase Sequences
1.	21239, (21239, 20655, 21242), (21239, 21242), (21366), (21242, 22246)
2.	(21239, 21366), 21242, (20655, 21242) , (21239, 21377)
3.	(21377, 22246), (21239, 20655), (21366, 22246), (21242,20655)
4.	21377, 22198 , (21239,22246), 21242, 20655, 21242
5.	(20674), (20674, 21245, 21239),21242
6.	21238,21239,21245
7.	(20655,20674,20675,20675),(21242,21245)
8.	20675,21238,21245
9.	21366,21239,21242,21245

to the timestamp. Table 5.4, then represents the sequences in the form of lists for learning products' vector representations using Prod2Vec.

Problem 1:

Given the input purchase sequence [‘20674’, ‘21242’, ‘20675’, ‘21245’, ‘20677’, ‘20655’] represented as a sentence with words (products) as $p_1 = 20674, p_2 = 21242, p_3 = 20675, p_4 = 21245, p_5 = 20677, p_6 = 20655$. Consider the product ‘20675’ as center product (word), the goal is to train the model to predict the neighboring (context) products which are [‘20674’ , ‘21242’ , ‘21245’, ‘20677’, ‘20655’] by learning vector representations. The model works as explained below.

Prod2Vec Algorithm Summary: Prod2Vec accepts product purchase sequences (e.g., as shown in Table 5.4) with some other input data listed above. It creates a product (word) vocabulary in the format of index: unique product id. Then, each product is represented in a one-hot vector format with dimension V where V is the vocabulary size and the po-

Table 5.4: Purchase sequences in list format

SID	Sequence(s) as Lists
S1	[21239, 21239, 20655, 21242, 21239, 21242, 21366, 21242, 22246]
S2	[21239, 21366, 21242, 20655, 21242 , 21239, 21377]
S3	[21377, 22246, 21239, 20655, 21366, 22246, 21242,20655]
S4	[21377, 22198 , 21239,22246, 21242, 20655, 21242]
S5	[20674, 20674, 21245, 21239,21242]
S6	[21238,21239,21245]
S7	[20655,20674,20675,20675,21242,21245]
S8	[20675,21238,21245]
S9	[21366,21239,21242,21245]

sition of the product is represented by a '1' in the vector. Next, a product of one-hot vector is taken with the input weight matrix X to get an embedded vector for the product which is then multiplied with the output weight matrix Y to get embedded vectors for the context products (e.g., as shown in Fig. 5.1). As a result, we obtain the Product Vector PV matrix (Nasir and Ezeife, 2020) where each product is represented across d=100 dimensions (features). Fig. 5.1 shows matrix PV for some sample products using Prod2Vec.

Figure 5.1: Product Vectors by Prod2Vec Model

$$PV_{p2vec} = \begin{bmatrix} & F0 & F1 & F2 & \dots & \dots & \dots & F97 & F98 & F99 \\ 20674 & 0.66072 & 0.176294 & -0.577179 & \vdots & \vdots & \vdots & 0.985326 & -0.113320 & 0.120507 \\ 21242 & 0.135658 & 0.208497 & -0.108157 & \vdots & \vdots & \vdots & 0.728440 & 0.023664 & 0.430904 \\ 20675 & 0.344165 & -0.012509 & -0.293663 & \vdots & \vdots & \vdots & 0.786582 & 0.182488 & 0.388695 \\ 21245 & 0.016946 & -0.051390 & -0.172432 & \vdots & \vdots & \vdots & 0.728440 & 0.023664 & 0.430904 \\ 20677 & 0.363226 & 0.076637 & -0.299294 & \vdots & \vdots & \vdots & 0.908400 & 0.081133 & 0.245601 \\ 20655 & -0.110575 & 0.226335 & -0.199941 & \vdots & \vdots & \vdots & 0.183031 & -0.334734 & 0.558633 \end{bmatrix}$$

We chose to use Prod2vec (Grbovic et al., 2015) to obtain product embeddings as it was the first approach to leverage the idea of word2vec (which was used to obtain word embeddings according to their context, i.e., neighboring words). Prod2vec (Grbovic et al., 2015) utilized the word2vec (Mikolov et al., 2013) approach by applying it to customers' historical records to obtain product embeddings.

Example 5.2. Learning Product Vector Representations Using Global Vectors (GloVe)

To use Glove model for our task, we represent a collection of purchase sequences consisting of product ID's sorted according to timestamp as documents and each product ID in the document representing a word. Glove will model the context of purchase sequences, where products with similar contexts that is, with similar neighboring purchases will have similar vector representations and are closer to each other.

The steps to learn product vector representations (semantics) by creating a co-occurrence matrix and then the vector representations using Glove are explained below:

Problem 2:

Given an input of purchase sequences (document) in a list of list format as [['20674', '21242', '20675', '21245', '20677', '20655'], ['20675', '21245', '20655'],['21245', '20674', '20675']] where a document consists of a collection of customers' purchase sequences sorted according to the timestamp and each sequence consists of one or more products (words) and a window size of n, the goal is to train the model to obtain the product co-occurrence matrix and then learn products' vector representations (semantics). Here in our example, we have three purchase sequences where each purchase sequence has a number of products represented by product ID's. The model works as explained below.

Input: Corpus of documents (purchase sequences in list of list format), window size - c .

Intermediates: Co-occurence matrix M of size (n x n) where n is the number of unique products in the sequences.

Output: Vector representation of all unique products in the vocabulary V across N dimensions. The created document (purchased sequences) will be fed to the model for creating co-occurrence matrix and then learning product vector representations.

Glove Algorithm Summary: Glove accepts product purchase sequences (e.g., shown in Table 5.4) with some other input data listed above. It creates a product (word) vocabulary in the format of index: unique product id. Then, it collects word (product) co-occurrence statistics to get a co-occurrence matrix (Fig. 5.2) so that it can compute the product co-occurrence score for each pair of products.

Steps:

1. Create a vocabulary of size (V) in the format index: product id consisting of all unique products. For example, $V = \{1 : 20674; 2 : 21242; 3 : 20675; 4 : 21245; 5 : 20677; 6 :$

20655}.

2. Collect word (product) co-occurrence statistics in a form of word (product) co-ocurrence matrix X. Each element X_{ij} of such matrix represents how often product i appears in context of product j. This is done by scanning the corpus as: for each product, look for context products within the defined window_size c (here c=1) after the term. Less weight is given for context products which are more distant from the target product, by using Eq.5.1:

$$\text{decay} = \frac{1}{\text{offset}} \tag{5.1}$$

FIGURE 5.2: Glove Model's Co-ocurrence Matrix

$$\left[\begin{array}{ccccccc} & 20674 & 21242 & 20675 & 21245 & 20677 & 20655 \\ 20674 & 0.0 & 1.0 & 1.5 & 1.33 & 0.25 & 0.20 \\ 21242 & 0.0 & 0.0 & 1.0 & 0.5 & 0.33 & 0.25 \\ 20675 & 0.0 & 0.0 & 0.0 & 2.5 & 0.50 & 0.83 \\ 21245 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 1.50 \\ 20677 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 20655 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{array}\right]$$

For example, in the above co-occurrence matrix, we can see that product "20674" (target) has co-occurred with product "21242" (context) as 1.0 times. This is computed by scanning the corpus and finding the positions where both these products occur together. Here, according to the corpus, they both occur together only in sequence 1 and the product "21242" is in the context window size (offset) of 1, so its co-occurrence score as per the Eq.5.1 is =1/1/= 1.0. In the same way the co-occurrence between products "20674" and "20677" (context) is computed. As product "20677" co-occurs with product "20674" in only one sequence (first sequence) and at an offset 4, so its co-occurrence score will be $= \frac{1}{4} = 0.25$. The co-occurrence score between other products is computed in the same way.

3. Produce vector values in continuous space for each word in the corpus, which represents how every pair of words i and j co-occur. This is done by using a soft constraint for each word pair of word i and word j which states that word vectors are learnt such that their dot product (inner product) equals the logarithm of the words' probability of co-occurrence as shown in Eq.5.2.

$$w_i^T w_j + b_i + b_j = \log\left(X_{ij}\right) \tag{5.2}$$

Here w_i represents vector for the target product, w_j is vector for the context product, b_i, b_j are scalar biases for the target and the context products. This is achieved by minimizing an objective function J, which evaluates the sum of all squared errors based on the above equation, weighted with a function f as given in Eq.5.3

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f\left(X_{ij}\right)\left(w_i^T w_j + b_i + b_j - \log X_{ij}\right) 2 \tag{5.3}$$

Where V is the size of the vocabulary. Here f is a weighting function which helps to prevent learning only from extremely common word pairs (product pairs). The following function (Eq.4) is used in Glove(Pennington et al., 2014):

$$f\left(X_{ij}\right) = \begin{cases} \left(\frac{X_{ij}}{x_{\max}}\right)\alpha, & \text{if } X_{ij} < XMAX \\ 1, & \text{otherwise} \end{cases} \tag{5.4}$$

By using the above formulas and the co-occurrence matrix obtained in step 3, we learn the feature vector representations of products as shown in Fig. 5.3.

$$PV_{glove} = \left[\begin{array}{cccccccccc} & F0 & F1 & F2 & \ldots & \ldots & \ldots & F97 & F98 & F99 \\ 20674 & -0.077740 & 0.075161 & 0.075892 & \vdots & \vdots & \vdots & -0.020273 & 0.133662 & 0.034656 \\ 21242 & -0.114833 & 0.061225 & 0.109533 & \vdots & \vdots & \vdots & -0.025963 & 0.173161 & 0.065046 \\ 20675 & -0.123344 & 0.098004 & 0.120447 & \vdots & \vdots & \vdots & -0.022385 & 0.189432 & 0.054446 \\ 21245 & -0.065362 & 0.047590 & 0.059333 & \vdots & \vdots & \vdots & -0.021115 & 0.114979 & 0.041389 \\ 20677 & -0.102480 & 0.087887 & 0.101183 & \vdots & \vdots & \vdots & -0.019029 & 0.167862 & 0.033685 \\ 20655 & -0.000398 & -0.002544 & 0.003422 & \vdots & \vdots & \vdots & -0.001759 & 0.001247 & 0.003912 \end{array}\right]$$

FIGURE 5.3: Product vectors by Glove model

Where each row represents feature vector representation of a product across d dimensions. Here, the feature vectors (dimensions) have a size of 100. This is a model parameter and can be set as required. For example, vector representation of product with id "20674" is the first row in the matrix and represented as $[-0.077740\ \ -0.075161\ \ 0.075892 \ldots\ldots\ldots -$ $0.020273\ \ 0.133662\ \ 0.034656]$.

Example 5.3. Learning Product Vector Representations Using TF-IDF

Problem 3:

Given a collection of documents d (e.g., books, articles, product descriptions, user reviews) and a set of features (terms) t, learn the document representation (semantics) by finding the association (weight) between the document and the features.
For our task, the problem is reformulated as:

Given a collection of documents (where a document represents collection of product descriptions in a list format as ['medium ceramic top storage jar', 'black candelabra t-light holder', 'woodland charlotte bag', 'airline bag vintage jet set brown'] where each element represents description of a product purchased and a set of features (where a feature will consist of unique tokens extracted from the product descriptions), find the association (weight) between the products and the features.

Input : $D = \{d_1, d_2, \ldots\ldots., d_N\}$, in our case,

D= ['medium ceramic top storage jar', 'black candelabra t-light holder', 'woodland charlotte bag', 'airline bag vintage jet set brown']

$T = \{t_1, t_2, \ldots\ldots, t_N\}$, in our case,

T= ['blue', 'bowl', 'dot', 'green', 'luggage', 'pink', 'plate', 'polka', 'queen', 'red', 'retrospot', 'skies', 'tag']

Output: $d_j =< w_{1j}, w_{2j}, \ldots., w_{nj}>$

TF-IDF Summary: TF-IDF aims to learn the product feature vectors by first creating set of all unique tokens (words) in product descriptions and then computing the term frequency count from the given product descriptions (Table 5.5 and Table 5.6). It then computes Inverse Document Frequencies (Table 5.7) and finally the product of Term frequency (TF) and Inverse document Frequency (IDF) is computed (Table 5.8 to obtain a term weighted vector representation, i.e., representing each product as an n-dimensional feature vector (Fig. 5.4). The steps to learn the product feature vectors using TF-IDF are given below:

Step 1: Term Frequency Computation (TF)

Table 5.5: Frequency Count of Unique Tokens occurring in the Product Descriptions

Prod Id	Products	Tokens												
		blue	bowl	dot	green	luggage	pink	plate	polka	queen	red	retrospot	skies	tag
20674	Green polka dot bowl	0	1	1	1	0	0	0	1	0	0	0	0	0
21242	Red retrospot plate	0	0	0	0	0	0	1	0	0	1	1	0	0
20675	Blue polka dot bowl	1	1	1	0	0	0	0	1	0	0	0	0	0
21245	Green polka dot plate	0	0	1	1	0	0	1	1	0	0	0	0	0
20677	Pink polka dot bowl	0	1	1	0	0	1	0	1	0	0	0	0	0
20655	Queen of skies luggage tag	0	0	0	0	1	0	0	0	1	0	0	1	1

Table 5.6: Term Frequencies (TF-Computation)

Product Id	Products	Tokens												
		blue	bowl	dot	green	luggage	pink	plate	polka	queen	red	retrospot	skies	tag
20674	Green polka dot bowl	0.00	0.25	0.25	0.25	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00
21242	Red retrospot plate	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.33	0.00	0.00
20675	Blue polka dot bowl	0.25	0.25	0.25	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00
21245	Green polka dot plate	0.00	0.00	0.25	0.25	0.00	0.00	0.25	0.25	0.00	0.00	0.00	0.00	0.00
20677	Pink polka dot bowl	0.00	0.25	0.25	0.00	0.00	0.25	0.00	0.25	0.00	0.00	0.00	0.00	0.00
20655	Queen of skies luggage tag	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.25	0.00	0.00	0.25	0.25

Table 5.7: IDF Computation

Terms	blue	bowl	dot	green	luggage	pink	plate	polka	Queen	red	retrospot	skies	Tag
IDF	0.78	0.30	0.18	0.48	0.78	0.78	0.48	0.18	0.78	0.78	0.78	0.78	0.78

Table 5.8: TF-IDF of Tokens in Product Descriptions (Product Vectors Using TF-IDF)

Terms	blue	bowl	dot	green	luggage	pink	plate	polka	Queen	red	retrospot	skies	Tag
IDF	0.78	0.30	0.18	0.48	0.78	0.78	0.48	0.18	0.78	0.78	0.78	0.78	0.78

Term Frequency (TF) will be computed by using 2.2. For example term frequency for "blue" in product description "green polka dot bowl" is computed as :
TF (blue, green polka dot bowl) $= \frac{0}{3} = 0$. Similarly, the term frequencies of all tokens are computed and shown in Table 5.5 which shows frequency count of thirteen terms occurring in the product descriptions For example, the term "bowl" appears three times among the product descriptions (as indicated by a 1 in each corresponding row where the term appears in a product).

Step 2: Inverse Document Frequency Computation (IDF)

IDF for all the terms is computed using the formula $log(\frac{N}{n_k})$ where N is the total number of product descriptions and n_k represents the number of product descriptions in which the term appears at least once. Table 5.7. shows the IDF of all terms.

Step 3: TF- IDF Computation

The TF-IDF is computed using 2.1 that is, taking the product of Term Frequency of each token in the product description with Inverse Document Frequency of the token in that product description. For example, the TF-IDF of the term blue will be:

TF-IDF (blue, green polka dot bowl) = TF (blue) * IDF (blue, green polka dot bowl)
=0.0 * 0.78 = 0.0
TF-IDF (bowl, green polka dot bowl) = TF (bowl) * IDF (bowl, green polka dot bowl)
= 0.25 * 0.30 = 0.08

After the computation of TF-IDF (Table 5.8), we can represent each product as an n-dimensional feature vector, where each dimension represents a feature (token). For our example, we have thirteen features (tokens) and the product "green polka dot bowl" can be represented as term weighted vector as: Green polka dot bowl = [0.00 0.08 0.04 0.12 0.00 0.00 0.00 0.04 0.00 0.00 0.00 0.00] After training the model on all products in the purchase sequence, we reduce the dimension of the features by using Singular Value Decomposition (SVD) and obtain final product vector matrix as shown in Fig. 5.4.

Example 5.4. Learning Product Vector Representations Using Doc2vec (Le and Mikolov)

FIGURE 5.4: Product Vectors obtained from TF-IDF Model

$$PV_{tfidf} = \left[\begin{array}{cccccccccc} & F0 & F1 & F2 & \dots & \dots & \dots & F97 & F98 & F99 \\ 20674 & 0.075502 & -0.026424 & -0.091291 & \vdots & \vdots & \vdots & 0.000088 & -0.073289 & -0.055986 \\ 21242 & 0.202582 & -0.226119 & -0.354920 & \vdots & \vdots & \vdots & 0.019935 & -0.028236 & 0.037867 \\ 20675 & 0.080445 & -0.041201 & -0.134280 & \vdots & \vdots & \vdots & 0.014696 & -0.030099 & -0.067652 \\ 21245 & 0.074386 & -0.036098 & -0.061312 & \vdots & \vdots & \vdots & 0.002169 & -0.076671 & 0.041561 \\ 20677 & 0.114951 & -0.034461 & -0.145326 & \vdots & \vdots & \vdots & -0.010915 & -0.047242 & -0.070353 \\ 20655 & 0.000589 & -0.001437 & -0.001532 & \vdots & \vdots & \vdots & -0.002298 & 0.004696 & 0.003122 \end{array}\right]$$

Problem 4:

Given a collection of documents d, learn the document vector representation.
For our task, the problem is reformulated as:

Given a collection of documents (where a document represents collection of product descriptions, product title and product brand in a list of list format as where each element represents description, title and brand of a product purchased, and a document ID for each document, learn document representation (product representation).

Input: A collection of documents in a list of list format [[green polka dot bowl earthenware largest measures hand wash tag limited], [red retrospot plate beautiful plates composed high rated heavy weight plastic material render plate leak free soak resistant cut proof unbreakable, silver spoon], [green polka dot plate add a splash of color bright party detail green white dots dessert plates party2u]].

Intermediates: vectors (word and and document ID vectors) with vector dimensions as $1 \times V$ (one-hot vector) and $1 \times N$ respectively, where N represents the total number of documents (product descriptions), weight matrix W from input to hidden layer and weight matrix Y from the hidden layer to the output layer.

Output: Vector representation of each document across N dimensions.

Doc2vec Summary and Steps: The method for learning document representation is similar to that of word2vec with the difference that along with the generation of word vector W for each word, a document vector D is also generated for each document during the training phase. For example, TaggedDocument (words=[‘green’ , ‘polka’ , ‘dot’, ‘bowl’ , ‘earthenware’, ‘largest’ , ‘measures’ , ‘hand’, ‘wash’, ‘tag’, ‘limited’] tags=[’0’]) and in the end of training, a numeric representation of the document (products) is represented as shown in Fig. 5.5.

$$PV_{Doc2vec} = \left[\begin{array}{cccccccccc} & F0 & F1 & F2 & \dots & \dots & \dots & F97 & F98 & F99 \\ 20674 & -0.729392 & -0.694623 & 0.360440 & \vdots & \vdots & \vdots & -0.059213 & 0.784198 & -0.709698 \\ 21242 & 0.478108 & 0.810839 & -0.67386 & \vdots & \vdots & \vdots & 0.876485 & 0.128663 & -0.818151 \\ 20675 & 1.560531 & 0.965368 & -1.731812 & \vdots & \vdots & \vdots & 0.182436 & -0.794381 & -0.585287 \\ 21245 & -0.309573 & 0.418730 & 0.247207 & \vdots & \vdots & \vdots & -0.468052 & 0.743325 & 0.427864 \\ 20677 & -0.100401 & 1.024751 & 0.034455 & \vdots & \vdots & \vdots & 0.346302 & -0.044198 & 0.284042 \\ 20655 & 0.685829 & 0.646765 & -0.305982 & \vdots & \vdots & \vdots & 0.438488 & 1.036193 & 1.161657 \end{array}\right]$$

FIGURE 5.5: Product Vectors by Doc2vec Model

We then created two hybrid matrices of product vector representations as (i) hybrid of PV_{p2vec} and PV_{glove} (Fig. 5.6 and (ii) hybrid of PV_{tfidf} and $PV_{doc2vec}$ (Fig. 5.7) to better learn the product semantics. This is achieved by averaging the embeddings of each product in the respective matrices.

$$PV_{p2vec\&glove} = \left[\begin{array}{cccccccccc} & F0 & F1 & F2 & \dots & \dots & \dots & F97 & F98 & F99 \\ 20674 & 0.005834 & 0.125728 & -0.250644 & \vdots & \vdots & \vdots & 0.482526 & 0.010171 & 0.077582 \\ 21242 & 0.010412 & 0.134861 & 0.000688 & \vdots & \vdots & \vdots & 0.351238 & 0.098413 & 0.247975 \\ 20675 & 0.110410 & 0.042747 & -0.086608 & \vdots & \vdots & \vdots & 0.382098 & 0.185960 & 0.221571 \\ 21245 & 0.024208 & -0.001900 & -0.056549 & \vdots & \vdots & \vdots & 0.483533 & 0.153193 & 0.265428 \\ 20677 & 0.130373 & 0.082262 & -0.099056 & \vdots & \vdots & \vdots & 0.444685 & 0.124498 & 0.139643 \\ 20655 & -0.055486 & 0.111896 & -0.098260 & \vdots & \vdots & \vdots & 0.090636 & -0.166744 & 0.281272 \end{array}\right]$$

FIGURE 5.6: Hybrid Product Vectors by Prod2vec and Glove Model

$$PV_{d2vecandtfidf} = \left[\begin{array}{cccccccccc} & F0 & F1 & F2 & \dots & \dots & \dots & F97 & F98 & F99 \\ 20674 & -0.034295 & 0.053539 & -0.088612 & \vdots & \vdots & \vdots & -0.024504 & -0.121241 & 0.018026 \\ 21242 & 0.042981 & -0.004462 & -0.270291 & \vdots & \vdots & \vdots & -0.024960 & -0.061913 & 0.104531 \\ 20675 & -0.022603 & 0.041644 & -0.109378 & \vdots & \vdots & \vdots & 0.004292 & -0.068960 & -0.068960 \\ 21245 & -0.021472 & 0.072360 & -0.129097 & \vdots & \vdots & \vdots & -0.064445 & -0.128279 & 0.069797 \\ 20677 & -0.007805 & 0.065704 & -0.117965 & \vdots & \vdots & \vdots & -0.014165 & -0.082256 & 0.027480 \\ 20655 & -0.038842 & 0.030226 & -0.069437 & \vdots & \vdots & \vdots & 0.097315 & 0.031924 & 0.204722 \end{array}\right]$$

FIGURE 5.7: Hybrid Product Vectors by Doctovec and TF-IDF Model

Chapter 6

Semantic Based Candidate Generation

In this Chapter, details are presented for generating potential candidate items for recommendation either by (a) mining semantic embedded sequential patterns and rules or (b) using Markov Model created from the daily purchase sequences of customers' historical purchase and enriched with proposed semantic transition probability matrix.
Section 6.1 presents details for generating candidate items using semantic based sequential pattern mining and Collaborative Filtering and Section 6.2 explains generating candidate items using semantic rich Markov Model along with walk through examples.

6.1 Semantics based Sequential Pattern Mining

This phase involves steps as (i) computing products' semantic similarity and creating item semantic similarity matrix, (ii) creating aggregated vector representation of purchase sequences, (iii) extracting Top-N semantically similar neighbors (users, items) and finally (iv) extracting semantic rich sequential patterns. Algorithm 3 presents the details.

Algorithm 3 Semantic Knowledge Integration into Sequential Pattern Mining Process (SKI_SPM)

```
Input         : Customers' purchase sequences (PS), Products' semantic feature vector
                matrix $PV_{hybrid}$, number of Top-N users with similar purchase behav-
                ior (N) as target customer, minimum support(s),minimum semantic sim
                (m_sem)
Output        : Semantic rich sequential purchase patterns
Intermediates: item semantic similarity matrix $M$, purchase sequence vector ($\overrightarrow{PS_u}$) of
                user $u$, similar purchase sequence data base (SPSDB)
1  Compute semantic similarity between products using cosine similarity (eq. 6.1) between
     their semantic feature vectors from hybrid $PV_{hybrid}$ matrix (section 6.1.1)
2  Item semantic similarity matrix ($M$) ← created using eq. 6.2 in section 6.1.1
3  for each purchase sequence (PS) do
4      $\overrightarrow{PS_u}$ ← calculate by taking average of vectors of all products in the purchase sequence
         using matrix $PV_{hybrid}$ (section 6.1.2)
5  for purchase sequence vector of each target user $\overrightarrow{PS_{ut}}$ do
6      for purchase sequence vector of each user $\overrightarrow{PS_u}$ do
7          compute similarity as similar users($\overrightarrow{PS_{ut}}$),$\overrightarrow{PS_u}$) as in section 6.1.3
8      sort results from the previous step   N ← store Top-N users with semantically similar
         purchases
9  SPSDB ← Extract purchase sequences of Top-N similar users from PS
10 SSP ← Generate semantic sequential patterns from SPSDB (section 6.1.4)
```

6.1.1 Computing Products' Semantic Similarity and item Semantic Similarity Matrix

Products' semantic similarity is computed by applying cosine similarity using Eq.6.1 on product vectors in the joint PV matrix.

$$\text{Cosine Similarity } (x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{6.1}$$

where x_i and y_i represent components of vectors for products x and y respectively. For example, to compute the similarity of product "20674" (Green Polka Dot Bowl) with product "21239" (Pink Polka Dot Cup) and product "20675" (Blue Polka Dot Bowl), we will take their corresponding product vectors (column) from the hybrid PV matrix and 21239, 20675 compute cosine similarity between them. So, cosine similarity, Cosine similarity (20674, 21239) is 0.81 and Cosine similarity (20674, 20675) is 0.98, which shows that product 20674 is more close to product 20675 in the vector space than product 21239. Similarity between other products is computed in the same way.

Cosine similarity was used as it measures the similarity between two vectors where in our case each product is represented in an n-dimensional vector space where each dimension represents products' semantic features obtained through product characteristics, concepts and context of usage (co-purchased, co-reviewed). Mathematically, Cosine similarity metric measures the cosine of the angle between two n-dimensional vectors which are projected in a multi-dimensional space. The Cosine similarity of two vectors will range from 0 to 1. If the Cosine similarity score is 1, it means two vectors have the same orientation. The value closer to 0 indicates that the two vectors have less similarity. Therefore the choice of cosine similarity fits to our requirement while computing similarities between product vectors.

Next, an item-item semantic similarity matrix M is populated with this computed product semantic similarity using Eq. 6.2. Each entry $R_{x,y}$ in the matrix M represents semantic similarity between products x and y in the vector space. Fig. 6.1. shows a sample of matrix M.

$$M_{x,y} = \begin{cases} 1, & \text{if } x = y \\ \text{CosineSimilarity } (x, y), & \text{otherwise} \end{cases} \tag{6.2}$$

FIGURE 6.1: Item to Item Semantic Similarity Matrix

$$M = \left[\begin{array}{cccccccccccc}
 & 20674 & 21242 & 21238 & 21245 & 21239 & 20655 & 20675 & 21366 & 22246 & 21377 & 22198 \\
20674 & 1.00 & 0.84 & 0.88 & 0.80 & 0.81 & 0.93 & 0.98 & 0.86 & 0.97 & 0.87 & 0.33 \\
21242 & 0.84 & 1.00 & 0.91 & 0.86 & 0.78 & 0.02 & 0.87 & 0.21 & 0.21 & 0.11 & 0.12 \\
21238 & 0.88 & 0.91 & 1.00 & 0.96 & 0.98 & 0.95 & 0.94 & 0.94 & 0.90 & 0.94 & 0.98 \\
21245 & 0.80 & 0.86 & 0.96 & 1.00 & 0.78 & 0.35 & 0.86 & 0.21 & 0.18 & 0.21 & 0.34 \\
21239 & 0.81 & 0.78 & 0.98 & 0.77 & 1.00 & 0.10 & 0.89 & 0.23 & 0.11 & 0.18 & 0.37 \\
20655 & 0.93 & 0.02 & 0.95 & 0.35 & 0.10 & 1.00 & 0.45 & 0.15 & 0.43 & 0.23 & 0.23 \\
20675 & 0.98 & 0.87 & 0.94 & 0.86 & 0.89 & 0.45 & 1.00 & 0.47 & 0.16 & 0.31 & 0.44 \\
21366 & 0.86 & 0.21 & 0.94 & 0.21 & 0.23 & 0.15 & 0.47 & 1.00 & 0.16 & 0.23 & 0.42 \\
22246 & 0.97 & 0.21 & 0.90 & 0.18 & 0.11 & 0.43 & 0.16 & 0.16 & 1.00 & 0.23 & 0.32 \\
21377 & 0.87 & 0.11 & 0.94 & 0.21 & 0.18 & 0.23 & 0.31 & 0.23 & 0.23 & 1.00 & 0.28 \\
22198 & 0.33 & 0.12 & 0.98 & 0.34 & 0.37 & 0.23 & 0.44 & 0.42 & 0.32 & 0.28 & 1.00
\end{array}\right]$$

6.1.2 Aggregate Vector Representation of Vector Sequences

This step involves computing vector representation (semantic information) of purchase sequences by aggregating vectors of each product in the purchase sequence. This is obtained by computing mean of vectors (across N dimensions) of all products in the purchase sequence by utilizing the hybrid matrices created in Chapter 5 Sect. 5.1.

6.1.3 Extracting Top-N Semantically Similar Neighbors

Next, Top-N neighbors with semantically similar sequences are identified by finding cosine similarity using Eq.6.1 between purchase sequence vector in the test data (target customer) and each purchase sequence vector in the training data. For example, to compute cosine similarity between the target customers' purchase sequence in the test data and other purchase sequences in the train data, consider a sample purchase sequence in test data as ['23077', '23078', '23076', '22437'] with vector representation $\overrightarrow{PS}_{ut}$ = [-0.044143 0.044143 0.001968 .... -0.001035 0.018966 0.02645] and a purchase sequence in training data [21238,21239,21245] with vector representation as $\overrightarrow{PS}_u$ = [0.032593 0.066846 -0.172250 .... -0.025199 -0.043636 0.08122], so, the cosine similarity between vectors of these two purchase sequences using Eq. 6.1 is 0.81.

Similarly, cosine similarity between this purchase sequence vector $\overrightarrow{PS}_{ut}$ in test data is computed with other purchase sequence vectors $\overrightarrow{PS}_u$ in the training data. The results of cosine similarity are then sorted in decreasing order to select Top-N (where N = 5, 10, or 15) customers with similar purchase behaviors, i.e., having products that are semantically similar to products in the target sequence. This process is repeated for all the purchase sequences in the test data (i.e., for each target user).

So, here we obtain semantically similar purchase sequences by computing similarities between the test and the train purchase sequences using the hybrid similarity matrices (PV_{p2vec_glove} and $PV_{doc2vec\_tfidf}$).

6.1.4 Mining Semantic Embedded Sequential Patterns and Rules

Purchase sequences of top-N semantically similar customers were extracted and database of those purchase sequences was created. Using Sequential Historical Database (SHOD) (Bhatta et al., 2019) format, these semantically similar purchase sequences were formatted to extract frequent semantic sequential patterns. For example, using SHOD format, a

Table 6.1: Purchase sequence database of Top-N customers

SID	**Sequence**
1	21239, (21239, 20655, 21242), (21239, 21242), (21366), (21242, 22246)
2	(21239, 21366), 21242, (20655, 21242) , (21239, 21377)
3	(21377, 22246), (21239, 20655), (21366, 22246), (21242,20655)
4	21377, 22198 , (21239,22246), 21242, 20655, 21242
5	(20674), (20674, 21245, 21239),21242
6	21238,21239,21245
7	(20655,20674,20675,20675),(21242,21245)
8	20675,21238,21245
9	21366,21239,21242,21245

sequence will be represented as <21239 21366-1 21242 -1 20655 21242 -1 21239 21377-2> where a -1 indicates the end of item and -2 indicates the end of sequence. For example, in our running example, Table 6.1, shows purchase sequences with products that are semantically similar to our target purchase sequence ['23077', '23078', '23076', '22437'].

Next, frequent semantic sequential purchase patterns from these semantically similar purchase sequences (Table 6.1 are extracted using Prefix Span (Pei et al., 2004). As these frequent sequences were generated from purchase sequences with products which are semantically similar, so the sequences obtained represents products which are similar in semantics and frequently purchased in sequential order. Detailed step by step example of extracting sequential patterns using PrefixSpan (Pei et al., 2004) is explained in Sect. 2.2.2.1.3. At this step, we enrich the process of mining sequential patterns by integrating semantic information of products from matrix M. The goal is to generate semantic rich frequent sequential patterns by pruning product sequences that have a semantic similarity score less than the specified similarity threshold in addition to the traditional method of removing sequences based on the support count. A similarity threshold of 0.5 was used. This gives us frequent sequential purchase patterns of products which are similar in semantics.
For example, first, some of the frequent sequential length-2 (L_2) itemsets with their support count are: L_2 = (21239, 20655):4, (21239, 21242):5, (21239, 22246):3, (20655, 21242):5, (21242, 20655):3, (21366, 21242):4. We will then check the semantic similarity between these products to reduce the search space and further prune the itemsets that are not semantically similar by looking at item to item semantic similarity matrix M(Fig. 6.1). In this case, the pattern (21239, 20655):4 and (21239, 22246):3 will be pruned out as semantic similarity between (21239, 20655) is 0.10 and between (21239, 22246) is 0.11 which are less than our

specified threshold of 0.5. So, final semantic sequential purchase patterns are:

L = 21239, 20655, 21242, 21366, 22246, 21377,21245, (21239, 21242), (21239,21245), (21242,21239), (21242,21245),(21239,21242,21239),(21239,21242,21245), (21242,21239,21242), (21242,21239,21245)

From these semantic sequential purchase patterns, one of the semantic sequential rule can be:

$$21239, 21242 \rightarrow 1245$$

Pink Polka Dot Cup, Red Retrospot Plate → Green Polka Dot Plate

Here we can see that the recommended product 21245 (Green Polka Dot Plate) is similar in semantics to the products 21239 (Pink Polka Dot Cup) and 21242 (Red Retrospot Plate) which the user has already liked or purchased and is more close to the interest of user.

We used Prefix Span (Jian Pei et al., 2001), as it extracts the sequential patterns through sequential by pattern growth method i.e., constructing projected databases with respect to the sequential pattern(s) (in which no candidate sequence needs to be generated). Furthermore, as the projected databases keep shrinking so it also lowers the computation process. The reasons is because in a sequence database, the number of sequential patterns that grow quite long, is usually small and therefore, in a projected database, when prefix grows, the number of sequences reduces substantially.

6.2 Semantic Enhanced Transition Probability Matrix in Markov Models

We propose to build a Markov Model from the daily purchase sequences of customers' historical purchase and consists of two steps which are (i) creating product pairs' sequential frequency matrix and (ii) creating transition probability matrix and then enriching this transition probability matrix with items' semantic knowledge. Algorithm 4 shows the steps and in the next subsequent sections, we will discuss each of these steps in detail.

Markov models satisfy the Markov property, i.e., the conditional probability distribution of future states depends only on the current state. In the simplest Markov model, known

Algorithm 4 Semantic and Sequential Knowledge Integration into Markov Model (SSKI_MM)

```
Input  : Customers' purchase sequences (PS), item semantic similarity matrix ($M$)
Output : Semantic Rich Transition Probability matrix $P_1$
1 Product Frequency ($PF$) matrix $\leftarrow$ computed based on the co-occurrence frequency of a
    product pair in customer purchase sequence (section 6.2.1)
2 Transition Probability matrix $P$ $\leftarrow$ computed by normalizing $PF$ matrix using Eq. 6.3
    (section 6.2.2)
3 Product pair score $\leftarrow$ computed based on semantic similarity and sequential relationship
    between a product pair using Eq. 6.4 (section 6.2.3)
4 Semantic rich Transition Probability matrix $P_1$ $\leftarrow$ created using Eq. 6.5 (section 6.2.4)
```

Table 6.2: Sample purchase sequences of customers

SID	Sequences
1.	(a,b),(a,c),(a,d),(a,e),(a,f),(c,b),(b,e)
2.	(b,a),(b,c),(b,d),(b,c),(b,e),(a,f),(c,b)
3.	(c,b),(a,c),(a,d),(a,e),(e,c)
4.	(e,b),(e,c),(a,e),(c,b),(b,e)

as first-order, each state is formed by a single action, i.e., a customer purchased a product. In the case of K-th-order models, the state-space will correspond to all possible sequences of K actions. As the available data could not adequately support the number of states of higher-order chains, these models would suffer from reduced coverage and possibly worse overall performance. Therefore, we adopted a first-order Markov chain.

We assume that the next-product purchase depends strongly on the purchase that was made recently or (the purchase which is happening now). Markov models are represented by the parameters $(S, \mathbf{P})$, where S is the set of states for which the Markov model is designed; and $\mathbf{P}$ is an $n \times n$ transition probability matrix, where n is the number of states (i.e., products in our example). In this context, state S_i is associated with the fact that the customer purchased the product i. Each entry $P_{i,j}$ corresponds to the probability of moving to state S_j when the process is in state S_i, i.e., purchasing product j after product i. Note that this matrix is not symmetric, i.e., $P_{i,j} \neq P_{j,i}$, as the order in which the products are purchased matters.

6.2.1 Product Pairs' Frequency Matrix Creation

Consider Table 6.2 which shows a sample purchase sequence database consisting of purchases made by customers sorted according to a time stamp. Each purchase sequence is assigned a sequence ID. For example, the purchase sequence with SID 2 that in the first transaction, the customer first purchased the product 'b' followed by the purchased of item 'a'. Then, in the next transaction (made on another timestamp), she purchased item 'b' followed by the purchase of item 'd' and so on. For the sake of simplicity we represent products using small alphabets a, b, c, d, e, f where 'a' = product with stock code 20674, 'b'= 21242, 'c' = 21238,'d'=21245, 'e'=21239, 'f' =20655 and 'g'='20675' (description of these products is already specified in Table 5.2 in Chapter 5, Sect. 5.1.

Based on the historical purchase information of the customers (purchase sequences from Table 6.2, we first compute Product Frequency, PF matrix, which is an $n \times n$ matrix that holds the counts of every pair of consecutive products purchased. Every pair of products (i, j) that a customer has taken consecutively is used to estimate the entry $pf_{i,j}$, i.e., the frequency of the event that state s_j follows the state s_i . The product frequency matrix PF of products from the purchase sequences is shown in Fig. 6.2. An entry $pf_{i,j}$ shows the frequency of the occurrence of the product pair i, j among all sequences. For example, the entry $pf_{a,b} = 0.25$ which shows that the purchase of product b followed by the purchase of product a occurs once among the four sequences so $pf_{a,b} = \frac{1}{4} = 0.25$. The "0" in the matrix shows that these pairs of products are never purchased together.

FIGURE 6.2: Product Frequency Matrix (PF)

$$PF = \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.25 & 0.50 & 0.5 & 0.75 & 0.25 & 0.25 \\ b & 0.5 & 0 & 0.25 & 0.25 & 0.75 & 0 & 0.5 \\ c & 0 & 1 & 0 & 0 & 0.25 & 0 & 0 \\ d & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ e & 0 & 0.25 & 0.5 & 0 & 0 & 0 & 0 \\ f & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Figure 6.3: Transition Probability Matrix P

$$\mathcal{P} = \left[\begin{array}{cccccccc} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & g \\ \text{a} & 0 & 0.11 & 0.22 & 0.22 & 0.33 & 0.11 & 0.1 \\ \text{b} & 0.22 & 0 & 0.11 & 0.11 & 0.33 & 0 & 0.22 \\ \text{c} & 0 & 0.8 & 0 & 0 & 0.2 & 0 & 0 \\ \text{d} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \text{e} & 0 & 0.33 & 0.66 & 0 & 0 & 0 & 0 \\ \text{f} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

6.2.2 Transition Probability Matrix Creation

After computing the frequencies of matrix PF, we need to normalize it to get $\mathbf{P}$, a row stochastic matrix (transition probability matrix), so that the total transition probability from state i to any other state will sum up to 1 according to Eq. 6.3 where the numerator represents the matrix entry at a row (starting state) and the denominator represents the sum of all states from that start state. For example, Fig. 6.3 shows the transition probability matrix $\mathbf{P}$ created after normalizing the product frequency matrix PF in Fig. 6.2.

$$\mathbf{P}_i = \frac{PF_i}{\sum_{j=1}^{n} PF_{i,j}}, \text{ if } \sum_{j=1}^{n} PF_{i,j} > 0 \tag{6.3}$$

6.2.3 Score Computation for Products

Next, enrich the transition probability matrix $\mathbf{P}$ with the semantic information of products (item similarities) from item to item similarity matrix M. This will be obtained by (i) computing a score for each product using Eq.6.4 and then (ii) updating the transition probability matrix entries by using Eq. 6.5.

$$\text{Score } (i,j) = \alpha(M_{i,j}) + \beta(\mathbf{P}(i,j)) \tag{6.4}$$

For example, Score between product pair (a, b) can be computed as:

$$\text{Score } (a,b) = \alpha(M_{i,j}) + \beta(\mathbf{P}(i,j))$$

$$= 0.8 * 0.84 + 0.2 * 0.11 = 0.69$$

Similarly,

Figure 6.4: Semantic and Sequentially rich updated Transition probability Matrix $\mathbf{P}_1$

$$\mathbf{P}_1 = \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.69 & 0.74 & 0.68 & 0.71 & 0.76 & 0.81 \\ b & 0.72 & 0 & 0.75 & 0.71 & 0.69 & 0.02 & 0.74 \\ c & 0.70 & 0.72 & 0 & 0.76 & 0.82 & 0.77 & 0.75 \\ d & 0.64 & 0.68 & 0.76 & 0 & 0.62 & 0.28 & 0.88 \\ e & 0.69 & 0.69 & 0.91 & 0.61 & 0 & 0.08 & 0.71 \\ f & 0.74 & 0.01 & 0.76 & 0.28 & 0.80 & 0 & 0.36 \\ g & 0.78 & 0.89 & 0.75 & 0.68 & 0.67 & 0.36 & 0 \end{array}\right]$$

Score$(a, d) = 0.8 * 0.80 + 0.2 * 0.22 = 0.68$ (where $\alpha = 0.8, \beta = 0.2$)
Similarly, score for other product pairs is computed.

6.2.4 Semantic and Sequentially Rich Transition Probability Matrix

The entries in the matrix $\mathbf{P}$ are updated with this score value showing the semantic and sequential relationship between products (by showing the transitioning probabilities from one state to another). Fig. 6.4 shows semantic and sequentially rich updated transition probability matrix $\mathbf{P}_1$ populated using Eq. 6.5 after score calculations for sample products. Matrix $\mathbf{P}_1$ can now be used to recommend next personalized semantic and sequential items to users. Here, the probability at being at the same state is 0, all diagonal entries are '0' (as we are assuming not to recommend the same product at time $t + 1$ which the customer has purchased in the current state at time t).

$$\mathbf{P}_1 = \begin{cases} 0, & \text{if } i = j \\ \text{Score } (i, j), & \text{otherwise} \end{cases} \tag{6.5}$$

This semantic and sequentially rich transition probability matrix also addresses the ambiguous prediction problem by having a different score measure (probabilities) of purchasing a particular product after the target product. For example, in Fig. 6.3., after the purchase of product a, products b and f have same probabilities which cause ambiguity as which product to select from both. However, this is resolved as shown in Fig. 6.4. where both products b and f have different probabilities stating that product f has higher probability of recommendation than product b. After normalization, we have the final semantic and sequential transition probability matrix as shown in Fig. 6.5.

$$\mathbf{P}_1 = \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array}\right]$$

Figure 6.5: Normalized semantic and sequentially rich Transition probability Matrix P

Chapter 7

Semantic & Sequential Next Item Recommendation

This chapter will explain the phase in the proposed system which generates semantic based sequential next item recommendations based on using any of the proposed semantic enhanced (a) Sequential Pattern Mining method (Algorithm 5), (b) Collaborative Filtering Method (Algorithm 6) or (c) Markov Model (Algorithm 7). The steps involved in this phase are (i) score computation for products, (ii) semantic and sequentially rich (a) item to item similarity matrix for Collaborative Filtering and (b) transition probability matrix for Markov Model and (iii) semantically rich and sequential Top-K item recommendation.

7.1 Recommendation Using Semantic Based Sequential Pattern Mining and Collaborative Filtering

In this section, details are presented to generate semantic and sequential next item recommendation using proposed semantic based (a) Sequential Pattern Mining (Algorithm 5) and (b) Collaborative Filtering (Algorithm 6).

Algorithm 5 Semantic and Sequential Next Item Recommendation using Semantic Sequential Pattern Mining (SSR_SPM)

```
Input  : Semantic rich sequential purchase patterns, min_conf, min_sup (s),item semantic
         similarity matrix ($M$), m_semsim
Output: Set of semantic and sequential rich Top-K recommended items (RS)
1  Generate rules from semantic rich sequential purchase patterns as
2  for each frequent sequential itemset $l$ do
3      generate all nonempty subsets of $l$
4  for every nonempty subset $s$ of $l$ do
5      $R \leftarrow$ output the rule $s \rightarrow (l - s)$ if support_count ($l$) / support_count($s$) >= min_conf,
         where min_conf is the minimum confidence threshold
6  for each item $i$ in user profile $u_p$ of target user $u_t$ do
7      if matching rule found in $R$ then
8          RS $\leftarrow$ match with rule antecedent and retrieve the rule consequent to create set of
             Top-K recommendations
9      else
10         if_no matching rule found in $R$ then
11         R $\leftarrow$ retrieve semantic and sequential score (i,j) from $M_1$ between item in $u_p$ and
             items in rule antecedent, store the rule where antecedent semantic similarity >=
             m_semsim and rule confidence is > = min_conf
12         RS $\leftarrow$ sort the results from RS and retrieve set of Top-K recommendations
```

Algorithm 6 Semantic and Sequential Knowledge Integration and Recommendation using Collaborative Filtering (SSKIR_CF)

```
Input        : Item semantic similarity matrix $M$, Semantic rich Sequential Patterns
               (SSP), number of Top-K recommendation items (K)
Output       : Set of semantic and sequential rich Top-K recommended items (RS)
1  Compute semantic and sequential score between a pair of products using Eq. 7.1 (section
     7.1.1)
2  Semantic and sequential rich item matrix $M_1 \leftarrow$ populated using Eq. 7.4 (section 7.1.2)
3  for each item $i$ in user profile $u_p$ of target user $u_t$ do
4      for each item $j$ in $M_1$ do
5          RS $\leftarrow$ retrieve score $m_{(i,j)}$ from $M_1$
6  RS $\leftarrow$ sort the results from RS and retrieve set of Top-K recommendations
```

7.1.1 Score Computation for Products

After extracting semantically rich frequent sequential purchase patterns (Ch. 6, Sect. 6.1, this sequential information about products is populated into the semantically rich item to item similarity matrix M. For each entry $R_{x,y}$ in the matrix M, we update the matrix entries by computing a score using Eq. 7.1,

$$\text{Score}(x,y) = \alpha(\text{ CosineSimilarity } (x,y)) + \beta(\text{ Confidence } (x,y)) + \gamma(\text{lift}(x,y)) \tag{7.1}$$

where CosineSimilarity(x, y) is already computed using Eq. 6.1 and Confidence (x,y) and lift(x,y) are computed using Eq. 7.2 and Eq.7.3 as:

$$\text{Confidence } (x,y) = \frac{\text{Support}(x,y)}{\text{Support}(x)} \tag{7.2}$$

$$\text{lift}(x,y) = \frac{\text{Support}(x,y)}{\text{Support}(x) * \text{Support}(y)} \tag{7.3}$$

where support(x,y) measures how frequently products x and y occur sequentially in all available sequences and Confidence(x,y) determines the sequential co-occurrence of products x and y given all sequences in which x occurs. The lift score lift (x,y) indicates whether there is a relationship between items x and y, or whether the two items are occuring together in the same order simply by chance (i.e., at random). Unlike the confidence metric whose value may vary depending on direction (e.g., confidence $(x \to y)$ may be different from confidence$(y \to x)$, lift has no direction. This means that the lift(x,y) is always equal to the lift (y,x).For example, consider the products "21242" and "21245", their cosine similarity with product "21239" is 0.58 and 0.88 respectively.
Confidence(x,y) and lift (x,y) based on their frequent sequential support count can be calculated as :

$$\text{Confidence } (21239, 21242) = \frac{\text{Support } (21239{,}21242)}{\text{Support } (21239)} = \frac{6/9}{7/9} = 0.85$$

$$\text{lift}(21239, 21242) = \frac{\text{Support}(21239{,}21242)}{\text{Support}(21239){*}\text{Support}(21242)} = \frac{6/9}{7/9{*}7/9} = 1.10$$

$$\text{Confidence } (21239, 21245) = \frac{\text{Support } (21239{,}21245)}{\text{Support } (21239)} = \frac{2/9}{7/9} = 0.28$$

$$\text{lift}(21239, 21245) = \frac{\text{Support}(21239{,}21245)}{\text{Support}(21239){*}\text{Support}(21245)} = \frac{2/9}{7/9{*}5/9} = 0.51$$

A lift score of 1 implies that there is no relationship between x and y and they occur together by chance). A lift score of greater than 1 shows a positive relationship indicating that x and y occur together more often whereas a lift score of less than 1 shows that both x and y occur together less often than random. Based on the above computed values, the Score(x,y) between products using Eq.7.1 can be computed as:

$$\begin{aligned}\text{Score}(21239, 21245) &= \alpha(\text{ CosineSimilarity } (x, y)) + \beta(\text{ Confidence } (x, y)) + \gamma(\text{ lift } (x, y) \\ &= 0.5 * 0.58 + 0.3 * 0.85 + 0.2 * 1.10 = 0.76\end{aligned}$$

Similarly,

$\text{Score}(21239, 21242) = 0.5 * 0.88 + 0.3 * 0.28 + 0.2 * 0.51 = 0.62$ where $\alpha + \beta + \gamma = 1$ Similarly, score for other products are computed.

7.1.2 Semantic and Sequentially Rich Item to Item Similarity Matrix

The entries in the matrix M are updated with this score value showing the semantic and sequential relationship between products. Fig.8.2.6.0.3 shows semantic and sequentially rich updated item to item matrix $M1$ populated using Eq.7.4 after score calculations for sample products. Matrix $M1$ can now be used by CF to recommend Top-K personalized items to users.

$$M1_{x,y} = \begin{cases} 1, & \text{if } x = y \\ \text{Score } (x, y), & \text{otherwise} \end{cases} \tag{7.4}$$

7.1.3 Semantically Rich and Sequential Top-K Recommendation

Finally, next item(s) for a user is predicted by taking the purchase sequence of each user in train data (user profile) and then generating recommendations for every item in the user profile by looking at its score with other available products from the matrix $M1$. Items having the highest score are retrieved and sorted in decreasing order. This process is repeated for all items in the user profile and then lists of top K items are generated which are semantically similar and purchased in sequential order.

In our running example, for the sequence where the user purchased products as <21242,

Figure 7.1: Semantic and Sequentially rich updated item to item matrix M1

$$M1 = \left[\begin{array}{cccccccccccc}
 & 20674 & 21242 & 21238 & 21245 & 21239 & 20655 & 20675 & 21366 & 22246 & 21377 & 22198 \\
20674 & 1.00 & 0.16 & 0.39 & 0.40 & 0.24 & 0.17 & 0.47 & 0.24 & 0.97 & 0.05 & 0.57 \\
21242 & 0.55 & 1.00 & 0.87 & 0.98 & 0.71 & 0.25 & 0.35 & 0.21 & 0.39 & 0.31 & 0.12 \\
21238 & 0.56 & 0.87 & 1.00 & 0.01 & 0.59 & 0.35 & 0.61 & 0.91 & 0.76 & 0.39 & 0.98 \\
21245 & 0.40 & 0.02 & 0.01 & 1.00 & 0.17 & 0.38 & 0.65 & 0.02 & 0.34 & 0.58 & 0.34 \\
21239 & 0.24 & 0.62 & 0.59 & 0.76 & 1.00 & 0.13 & 0.57 & 0.42 & 0.41 & 0.15 & 0.37 \\
20655 & 0.17 & 0.21 & 0.35 & 0.38 & 0.13 & 1.00 & 0.07 & 0.69 & 0.59 & 0.61 & 0.23 \\
20675 & 0.98 & 0.87 & 0.94 & 0.65 & 0.86 & 0.07 & 1.00 & 0.42 & 0.13 & 0.28 & 0.44 \\
21366 & 0.86 & 0.21 & 0.94 & 0.02 & 0.67 & 0.62 & 0.42 & 1.00 & 0.47 & 0.59 & 0.42 \\
22246 & 0.97 & 0.21 & 0.90 & 0.34 & 0.41 & 0.04 & 0.90 & 0.47 & 1.00 & 0.15 & 0.32 \\
21377 & 0.87 & 0.11 & 0.94 & 0.58 & 0.15 & 0.96 & 0.28 & 0.59 & 0.15 & 1.00 & 0.28 \\
22198 & 0.33 & 0.12 & 0.98 & 0.39 & 0.67 & 0.57 & 0.76 & 0.90 & 0.73 & 0.66 & 1.00
\end{array}\right]$$

Table 7.1: Product recommendation using semantically enhanced Sequential Pattern Mining and Collaborative Filtering Matrix

User's Purchases Products	Top-3 Products (Semantically similar and Sequential)
21242	21245 21238 21239
20655	21242 20655 20675
20675	20674 21239 21238

20655, 20675>, the Top-K (K=3) semantic and sequentially similar products to each of the product in the purchase sequence which are recommended to the user are shown in Table 7.1.

Eliminating the common products and those already purchased by the user, the final set of recommended items for the user will be $\{21245, 20674, 21238, 21239\}$.

7.2 Recommendation Using Semantic Enhanced Transition Probability Matrix in Markov Model

In this subsection, details are presented to generate semantic and sequential next item recommendation using proposed Markov Model which is based on semantic enhanced transition probability matrix (Algorithm 7).

For next item recommendation, a Markov Model can be traversed as a random walk. A random walk on a directed graph consists of a sequence of vertices generated from a start vertex by selecting an edge, traversing the edge to a new vertex, and repeating the

Algorithm 7 Recommendation using Semantic Rich Transaction Probability Matrix in Markov Model (SSR_MM)

```
Input     : Semantic rich Transaction Probability matrix $P_1$, Customers' purchase
            sequece (PS), number of steps ($L$)
Output    : Probability Vector for Next Item Recommendation at time t+1 $P^{t+1}$
1  Consider Markov Model's semantic rich Transition Probability matrix as an item to item
     graph, where each item in the matrix represents a vertex and each entry $p_{i,j}$ represents
     probability of walk at vertex $i$ selecting the edge to vertex $j$
2  Initiate the walk at vertex $x_0$ (item in user's purchase sequence) at time $t$ by computing
     a starting probability distribution vector $p^t$ (representing the probability of initiating the
     walk at vertex $i$ (item in the purchase sequence)
3  if user purchase sequence has single item at time step $t$ then
4      $p^t \leftarrow$ computed using Eq. 7.5 (Section 7.2)
5      Next items' probability vector $p^{t+1}$,after a single time step ($L$), at time $t$+1 $\leftarrow$ computed
         using Eq. 7.6 (Section 7.2, Example 7.1)
6      Next items' probability vector $p^{t+1}$,after multiple time steps ($L$), at time $t+1 \leftarrow$ com-
         puted using Eq. 7.7 (Section 7.2, Example 7.1)
7  else
8      if user purchase sequence has multiple items at time step $t$ then
9          Each item (vertex) will have equal probability to begin the walk
10         $p^t \leftarrow$ computed using Eq. 7.5 (Section 7.2)
11         Next items' probability vector $p^{t+1}$, after a single time step ($L$), at time $t+1 \leftarrow$
             computed using Eq. 7.6 (Section 7.2, Example 7.2)
12         Next items' probability vector $p^{t+1}$,after multiple time steps ($L$), at time $t+1 \leftarrow$
             computed using Eq. 7.7 (Section 7.2, Example 7.2)
```

process. So, in this case, a Markov Model is considered as an item-to-item graph based on the transition probability matrix where each item in the matrix represents a vertex and each entry $p_{i,j}$ represents probability of the walk at vertex i selecting the edge to vertex j. To initiate the walk at a vertex x_o at time t, a starting probability distribution is required and then its product with the transition probability matrix is computed to obtain the probability of being at vertex x at time $t+1$ (that is , the probability of going from state i at time t to state j at time $t+1$). The initial probability distribution represented as $\mathbf{p} \in \mathcal{R}^{1\times n}$ at time t, where $\mathbf{p}$ is a row vector with non-negative components whose sum equals 1, and $\mathbf{p}_x$ being the probability of starting at vertex x, $\mathcal{R}$ representing set of non-negative numbers.

The initial probability distribution at time t can be computed (i) by starting the walk at a given vertex or (ii) initiating the walk at random (where all vertices have an equally likely chance of being selected). For example, if we have states (vertices) as $\{a, b, c, d, e, f, g\}$ and we want to initiate the walk at a given vertex c, then the starting probability distribution vector $\mathbf{p}$ will be $\mathbf{p}$ = [0 0 1 0 0 0 0] i.e., the vertex from where we will initiate the walk will have '1', i.e., $\mathbf{p}_x = 1$ where $x = c$ and remaining vertices will have '0'. However, to start the walk at random from any vertex at time t where each vertex is equally likely to be selected to initiate the walk, the initial probability distribution for each vertex will be $\frac{1}{degree(v)}$, where degree(v) represents the number of vertices. For example, here it will be $\mathbf{p}_x$ =[$\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$ $\frac{1}{7}$].

In our setting, the random walk for customer j will equally start from any given product in the customers' last purchase, so the initial probability distribution will be computed using Eq. 7.5. This also serves as a customer's personalization vector as we are taking into account customers' purchase history for each customer contrary to other approaches where same state of initial probabilities is used for all customers to determine the transition probabilities to the next state.

$$\mathbf{p}^t = \begin{cases} 1/\left|I_{j,tj}\right| & \text{if } i \in I_{j,tj} \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

Where $I_{j,tj}$ is the set of all items purchased at time t. Now, given an initial probability distribution $\mathbf{p}^t$ which is a row vector with a component for each vertex specifying the probability of the vertex at time t and the transition probability matrix $\mathbf{P}_1$, we can walk over the Markov Model to obtain $\mathbf{p}^{t+1}$ (the row vector of probabilities at time $t+1$ using Eq. 7.6 which represents the probabilities of transitioning to next state j at time $t+1$. Next, we present two example usecases to understand the next item recommendation process.

$$\mathbf{p}^t\mathbf{P} = \mathbf{p}^{t+1} \tag{7.6}$$

Example 7.1: Semantic and Sequential Next Item Recommendation with Single item purchase at time t

For example, if the customers' last purchase sequence contains product <c>, we want to predict (recommend) the next item after the purchased item 'c', then our initial probability distribution vector will be $\mathbf{p}^t$ =[0 0 1 0 0 0 0]. Using Eq. 7.6 we take product of the enriched transition probability matrix and the initial probability vector to get $\mathbf{p}^{t+1}$ as :

$$\mathbf{p}^{t+1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array} \end{bmatrix}$$

$$= \begin{bmatrix} 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \end{bmatrix}$$

So, the next product recommended to customer will be product ‘e’ which has high score of 0.181.

Furthermore, if we want to compute the probabilities (recommend items) after $t+1$ steps, it will be obtained by the sum over each adjacent vertex i of starting at i and taking the transition from i to j using Eq.7.7,

$$\mathbf{p}\mathbf{P}^{t} = \mathbf{p}^{t+1} \tag{7.7}$$

For example, if we want to predict the next possible purchase after 3 time steps (i.e., after purchase of three products where each purchase involves transition from one product to the next). So, substituting values in the above equation we get, $\mathbf{p}P^{3} = \mathbf{p}^{4}$ as:

Table 7.2: Product recommendation by proposed semantic enhanced Transition Probability Matrix in Markov method

User's last Purchased Product	Recommended Products (Semantically similar and Sequential)		
	t=1	t=2	t=3
c (Red retrospot cup)	e (Pink polka dot bowl)	a (Green polka dot bowl)	c (Red retrospot cup)
a (Green polka dot bowl)	g (blue Polka dot bowl)	a (Green polka dot bowl)	c (Red retrospot cup)
b (Red retrosport plate)	a (Green polka dot bowl)	b (Red retrosport plate)	c (Red retrospot cup)
e (Pink polka dot bowl)	c (Red retrospot cup)	e (Pink polka dot bowl)	c (Red retrospot cup)

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \left(\begin{bmatrix} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{bmatrix} \right)^3$$

$$= \begin{bmatrix} 0.151 & 0.145 & 0.160 & 0.141 & 0.150 & 0.091 & 0.158 \end{bmatrix}$$

So, the recommended product after time 3 (after 3 purchases) will be product c as it has the highest probability. So, the recommended sequence of customer purchase will be <e, a, c>.

In our running example, for the users where the last purchased product is c, the recommended products upto 3 time steps will be e, a and c as shown in Table 7.2. Results are shown for some other products as well.

The recommended products results show that the probability of the walker to reach the vertices after K steps provides an intuitive measure that can be used to rank the products and offer personalized recommendations to the customers accordingly.

Example 7.2: Semantic and Sequential Next Item Recommendation with multiple items purchase at time t

For example, if the customers' last purchase sequence contains products < bac>, we want to predict (recommend) the next item after the purchase, then our initial probability distribution vector will be $[\frac{1}{7}\frac{1}{7}\frac{1}{7}0000]$, we take product of the enriched transition probability matrix and the initial probability vector using Eq.7.6 to get $\mathbf{p}^{t+1}$ as :

$$\mathbf{p}^{t+1} = \begin{bmatrix} \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & 0 & 0 & 0 & 0 \end{bmatrix} * \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array}\right]$$

$$= \begin{bmatrix} 0.0494 & 0.0447 & 0.0518 & 0.0720 & 0.0744 & 0.0484 & 0.0784 \end{bmatrix}$$

So, the next product recommended to customer will be product g which has high score of 0.078. Similarly, to recommend products after $t+1$ steps (e.g., after 3 steps), we use Eq.7.7 to get, $\mathbf{pP}^3 = \mathbf{p}^4$

$$\begin{bmatrix} \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & 0 & 0 & 0 & 0 \end{bmatrix} * \left(\left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array}\right] \right) 3$$

$$= \begin{bmatrix} 0.0646 & 0.0587 & 0.0590 & 0.0680 & 0.0640 & 0.0361 & 0.0666 \end{bmatrix}$$

So, the recommended product after 3 steps will be d as it has the highest probability and the recommended sequence of customer purchase will be <g, c, d>.

In our running example, for the users where the last purchase sequence is <bac>, the recommended products upto 3 time steps will be g, c and d as shown in Table 7.3. Results are shown for some other products as well.

TABLE 7.3: Product Recommendation by our proposed method

User’s last Purchased Product	Recommended Products (Semantically similar and Sequential)		
	t=1	t=2	t=3
<b,a,c> (Red retrospot plate, Green polka dot bowl, Red retrospot cup)	g (Blue Polka dot bowl)	c (Red retrospot cup)	d (green polka dot plate)

Chapter 8

Experiments and Analysis

In this chapter, we will present details for our experiments conducted to evaluate the proposed system when (i) semantic integrated sequential patterns and semantic enhanced CF models are used (Section 8.1) and (ii) semantic enhanced transition probability matrix is used in Markov Model (Section 8.2) for generating personalized semantic and sequential next item recommendation.

8.1 Evaluation of Proposed System with Semantic Integrated Sequential Patterns and Collaborative Filtering Method

8.1.1 Datasets and Implementation Details

- Online Retail[1] : This dataset contains purchases made during an eight-month period between 01/12/2010 and 09/12/2011 for a UK based retail company that sells unique all-occasion gifts.
- Amazon[2] : This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). To test our model, we selected the review-K core (which is a subset of the data set where all items have at least K reviews, where

[1]https://archive.ics.uci.edu/ml/datasets/online+retail
[2]http://jmcauley.ucsd.edu/data/amazon/

K=5) and product metadata for categories including fashion, beauty, movies and TV. Details of data set statistics are provided in Table 8.1.

We implemented the proposed model using python. For mining sequential patterns using PrefixSpan (Jian Pei et al., 2001), we used Open source data mining library SPMF[3] . To compare our proposed model with the baselines we used the code provided from their authors and the github repository[4].

8.1.2 Pre-processing and Hyper-parameter Tuning

For data set partitioning, we adopted commonly used strategies of (i) leave one out (the most recent, i.e., last sequence of each user is used for testing and all remaining sequences for training) and (ii) temporal user splitting (where a percentage of the last interactions of each user is reserved for testing rather than just one). Availability of a rating or review (Amazon) and purchase (Online Retail) is considered as user-item interaction and we used timestamps to determine the sequential order of actions. Purchases made by each customer were grouped into sequences according to the timestamp. Data was pre-processed to create train and test data. For leave-one-out, the training data was created from those purchase sequences and the last purchase sequence of each customer was used to create the test set for evaluating model's performance. In the temporal user splitting, we used train and test splits of (a)70%, 30% and (b) 80%, 20%. Users with at least 5 purchasing records are selected.

All the models have some parameters to tune. We follow the reported optimal parameter settings for the baseline methods. For our model, the embedding dimension d is determined by grid search in the range $\{10, 20, 30, 40, 50, 100\}$, number of Top users with similar behavior N as $\{5, 10, 15\}$, number of Top recommendation items K in $\{1, 5, 10, 20, 50, 100\}$, minimum support s (%) for mining sequential patterns as $\{1, 2, 3\}$ and semantic similarity m_sem=0.5. The values of coefficients (alpha, beta and gamma) while computing score measure for products were explored through grid search and the best results were

[3]https://www.philippe-fournier-viger.com/spmf/index.php
[4]https://github.com/mquad/sars tutorial

Table 8.1: Data set statistics (product reviews and meta data)

		Reviews (Amazon) and Purchases (Online Retail) Data							**Meta Data**
Data set /Statistics		**Total no. of Transactions**	**No. of Unique Users**	**No. of Unique Items**	**Avg. no. of Reviews/ Purchases per item**	**Max. Sequence length**	**Avg. Sequence length**	**Min. Sequence Length**	**No. of Unique Items**
Amazon	Fashion	883636	743216	186054	4.66	40	1.17	1	186637
	Movies and TV	8290109	3755907	181996	45.55	4036	2.21	1	203970
	Beauty	353956	317982	32586	10.86	23	1.11	1	32992
Online Retail		240007	2974	3282	58.10	294	16.64	1	4497

with $\alpha = 0.5, \beta = 0.3$ and $\gamma = 0.2$. For other parameters, optimal performance was with $d = 100, N = 15ands = 1$, m_sem=0.5.

8.1.3 Evaluation Metrices

The model was evaluated on many metrices including Precision@K, Recall@K, Mean Reciprocal Rank (MRR), Hitrate@K and NDCG@K as used in (Kang and McAuley, 2018b, Tang and Wang, 2018, Wang and Kadıoğlu, 2021). For a given length of user profile in a test sequence, we predict a list of Top-K items denoted as $\hat{R}_{1:K}$ and the remaining part of the test sequence i.e., ground truth denoted as R. The different evaluation metrices are defined as:

- **Precision@K:** It is defined as the proportion of recommended items in the top-K set that are relevant and computed as:

$$\text{Precision@K } = \frac{|R \cap R^{\wedge}_{1:K}|}{K} \tag{8.1}$$

- **Recall@K:** Recall is defined as the proportion of relevant items found in the top-K recommendations.

$$\text{Recall@K} = \frac{\left|R \cap \hat{R}_{1:K}\right|}{|R|} \tag{8.2}$$

- **Mean Reciprocal Rank (MRR):** The reciprocal rank of items recommended is the multiplicative inverse of the rank of the first correct recommended item among the top-K recommendations. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of ground truths R, where $rank_i$ refers to the rank position of the first relevant item for the i-th ground truth. It is computed as:

$$MRR = \frac{1}{|R|} \sum_{i=1}^{|R|} \frac{1}{\text{rank}_i} \tag{8.3}$$

- **Hit rate@K:** Percentage of users that can receive at least one correct recommendation.

- **Normalized Discounted Cumulative Gain (NDCG@K):** Evaluates ranking performance by taking the positions of correct items into consideration. NDCG@k is normalized to [0, 1] and a perfect ranking is represented by 1. For each user, the NDCG is computed using the following,

$$\text{NDCG@ K} = \frac{\sum_{k=1}^{K} \frac{I_{c_j}^{k}}{\log_2(k+1)}}{\sum_{k=1}^{m_{c_j}^{K}} \frac{1}{\log_2(k+1)}} \tag{8.4}$$

where $I_{u_i}^{k}$ will be 1 if the kth recommendation for customer c_j is relevant, or it exits in the actual response, and 0 otherwise. Besides, $m_{c_j}^{K}$ is the number of relevant items for customer c_j up to the Kth recommendation. We used the average of NDCG over all users as the final metric of a method. For each user in a test sequence, we predict lists of Top-K personalized items where K is in $\{1, 5, 10, 20, 30, 50, 100\}$. We first computed the per-user score for each K and then reported the global average score for all users for each K.

8.1.4 Complexity Analysis

The dominant term in the computational complexity of our proposed model is $\mathcal{O}(n^2)$ mainly due to computing similarity at item level and sequence level. Next, we discuss the complexity according to the models, data processing and mining of sequential patters.

8.1.4.1 Complexity of Models

- Prod2vec: The complexity is proportional to the vocabulary size (unique products), which is computationally expensive in practical tasks as it can easily reach millions of products. As an alternative, negative sampling is used which significantly reduces the computational complexity. Depending on the size of the corpus, the complexity grows linearly with the size of the corpus. Therefore, if N represents the size of the corpus and V represents the unique products in the vocabulary, the complexity will be $\mathcal{O}(N * log(V))$.
- Glove: The computational complexity of the model depends on the number of nonzero elements in the co-occurrence matrix X. As this number is always less than the total number of entries of the matrix, the model scales no worse than $\mathcal{O}(|V^2|)$, where V represents the size of the vocabulary.

- Doc2vec: The runtime of the model is linear in the number of input documents, i.e., purchase sequences with products and meta data. Keeping all other parameters equal, increasing the number of input documents, the runtime will be increased.
- TF-IDF: If N represents the number of documents and TN represents the total number of terms, then the worst-case time complexity of this will be $\mathcal{O}(TN * N)$. In practice, the number of documents in which a particular term appears is very less and hence the time taken will be much lower than that.

8.1.4.2 Complexity of dataset Pre-processing

The various Natural Language Processing (NLP operations were performed using the Natural Language ToolKit (NLTK) library from python which consists of a set of text processing libraries for faster data pre-processing.

8.1.4.3 Complexity of Mining the Sequential Patterns

As we used Prefix Span (Jian Pei et al., 2001), in which no candidate sequence needs to be generated so major complexity is involved in constructing projected databases with respect to the sequential pattern(s). However, as the projected databases keep shrinking so it also lowers the computation process. The reasons is because in a sequence database, the number of sequential patterns that grow quite long, is usually small and therefore, in a projected database, when prefix grows, the number of sequences reduces substantially. In the worst case, in PrefixSpan, a projected database is constructed for every sequential pattern. If there are N sequential patterns then the complexity for constructing projected databases will be $\mathcal{O}(N)$. The cost is non-trivial, if a good number of sequential patterns exists.

8.1.5 Baseline Methods for Comparison

To show the effectiveness of our model, we considered recommendation baselines under three groups.

The first group includes general recommendation methods based on user feedback without any sequential order of user actions.

1. **Popularity Based (POP).** All items are ranked by their popularity in all users' sequences, where popularity is determined by the number of interactions.

2. **Baysian Personalized Ranking (BPR) (Rendle et al., 2009)**. A state of the art method for non-sequential item recommendation on implicit feedback, utilizing Matrix Factorization Model.

   The second group includes sequential recommendation methods based on sequential pattern mining and first order Markov chains, which consider the last visited item.

3. **Historical Purchase Click (HPCRec) (Xiao and Ezeife, 2018) and Historical Sequential Purchase (HSPRec) (Bhatta et al., 2019)**. It mines frequent sequential click and purchase behavior patterns using the consequential bond between click and purchase sequences and then using this quantitatively and qualitatively rich matrix for collaborative filtering to provide better recommendations.

4. **Factorized Personalized Markov Chain (FPMC) (Rendle et al., 2010)**. A hybrid approach that combines matrix Factorization (MF) which factorizes the matrix on user-item preferences for learning users' general taste and Markov Chains(MC) that models sequential behavior through a transition graph built over items which predicts users' next action based on the recent actions.

   The third group includes sequential recommender systems based on deep learning, which include various or all previously visited items.

5. **GRU4Rec (Hidasi et al., 2016c)**.To model sequential dependencies and making predictions in session based recommendation systems, proposed this method based on Recurrent Neural Networks (RNN's).

6. **Convolutional Sequence Embeddings Caser (Tang and Wang, 2018)**. A convolutional neural network (CNN) based method which takes the embedding matrix of the L most recent items and applies convolution operations on it to achieve sequential recommendation.

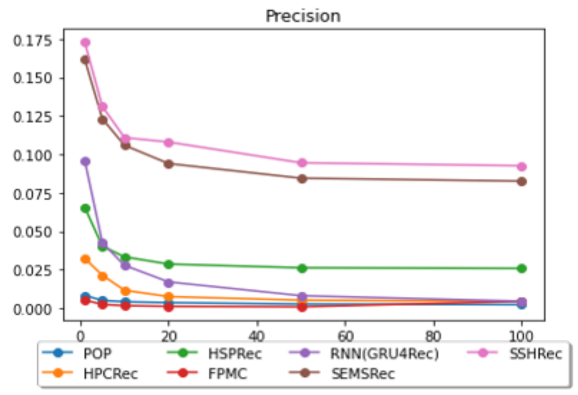
7. **Self Attentive Sequential (SASRec) (Kang and McAuley, 2018b)**. It captures long term user preferences by using attention mechanism and makes its predictions based on relatively few actions.

8.1.6 Results and Analysis

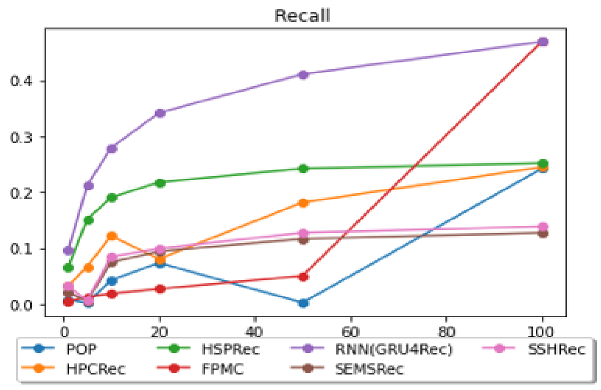
Our proposed model SSHRec (Nasir et al., 2021) gave improved performance after incorporating products' meta data to learn product semantics and using semantic similarity, confidence and lift measures to compute relationship between products in comparison to our previously proposed SEMSRec (Nasir and Ezeife, 2020) model (without using any product data) and other baselines on all K tested. High precision and recall measures of sequential recommenders such as SASRec (Kang and McAuley, 2018b), Caser (Tang and Wang, 2018), HSPRec19 (Bhatta et al., 2019) and RNN (Hidasi et al., 2016c) indicate that learning sequential information about customers' behaviour is important to capture user's long and short term preferences and to improve quality of recommendations. Factorized Personalized Markov Chain (FPMC) (Rendle et al., 2010) model on the other hand, showed least performance with the lowest precision and recall score, indicating that it could not learn the semantics of items and sequential purchase patterns of customers effectively. SEMSRec (Nasir and Ezeife, 2020) on the other hand gave slightly high performance as compared to other sequential recommender baselines indicating that integrating items' semantic information to compute item similarities can improve the quality of recommendations. However, the proposed extension SSHRec has outperformed SEMSRec significantly showing the importance of using meta data for learning product semantics and using semantic similarity, confidence and lift measures to compute relationship between products. Fig. 8.1 shows the performance of SSHRec in comparison to other systems on various evaluation metrices.

Furthermore, the results of the proposed model (SSHRec) on different datasets including Online Retail and various categories in Amazon data (Fashion, Beauty, Movies) are presented in Table 8.2. Here, we report results on all evaluation metrics at a cutoff of K=10. We notice that SSHRec performed considerably well Amazon's beauty dataset with a hitrate of 67% and 45% respectively.

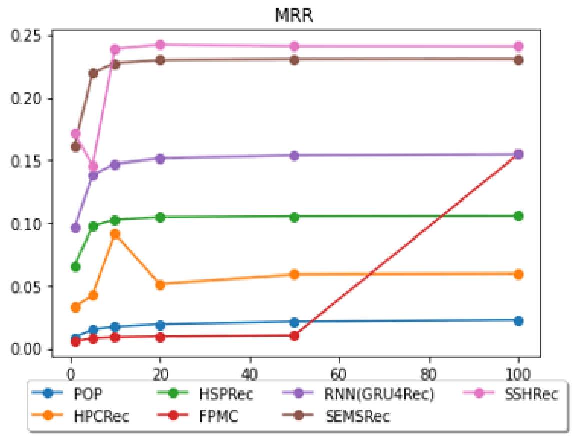
Next, we will discuss the impact of the parameters, minimum support, s and number



(A) Precision



(B) Recall



(C) MRR

FIGURE 8.1: Performance comparison of proposed SSHRec with other models

of customers with similar purchase behavior (N) one at a time by holding the remaining parameters at their optimal settings.

8.1.6.0.1 Influence of Top-N customers (N) We vary the number of customers N with similar purchase behaviors generated after finding semantic similarity between training sequences and the target sequence, to explore how it effects the quality of overall recommendations. We used different values of N as 5, 10 and 15 respectively. By increasing the value of N, gradual decrease in model performance was noticed in terms of Top-K recommendations. This is because when the number of customers is increased, number of similar purchase sequences also increases, so we do get more similar products to recommend from, however, those similar products are not necessarily purchased in sequence (which is important to capture users' long and short term behaviors) which lowers the recommender's performance. Optimal performance was when $N = 5$. Futhermore, choosing N less than five yielded purchase sequences which were very short in length and did not contribute much in yielding useful semantic sequential purchase patterns and rules therefore we did not consider that.

Table 8.2: Results of Proposed System SSHRec on different datasets with K=10

Evaluation Metrices/Data Set	**Amazon**			**Online Retail**
	Fashion	Movies and TV	Beauty	
Precision@K	0.0075	0.0165	0.0909	0.1006
Recall@K	0.0194	0.0360	0.4856	0.1145
MRR	0.0126	0.0276	0.1705	0.1509
HitRate@K	0.0559	0.1326	0.6705	0.4544
Mean Average Precision	0.0075	0.0154	0.0855	0.0887
NDCG@K	0.0166	0.0319	0.3696	0.1450

8.1.6.0.2 Influence of Embedding size We vary the size of embeddings d and trained multiple models based on embedding size $\{10, 20, 30, 50, 100\}$ and found that $d = 100$ gave optimal results. The detailed results are shown in Table 8.3.

8.1.6.0.3 Influence of train and test split For data set partitioning, we adopted the strategies of (i) leave one out (the most recent, i.e., last sequence of each user is used for testing and all remaining sequences for training) and (ii) temporal user splitting (where a percentage of the last interactions of each user is reserved for testing rather than just one). Availability of a rating (Amazon) and purchase (Online Retail) is considered as user-item interaction and we used timestamps to determine the sequential order of actions.

Data was preprocessed to create train and test data. Purchases made by each customer were grouped into sequences according to the timestamp. For leave-one-out, the training data was created from those purchase sequences and the last purchase sequence of each customer was used to create the test set for evaluating model's performance. In the temporal user splitting, we used train and test splits of (a)70%, 30% and (b) 80%, 20%. Users with at least 5 purchasing records are selected. The experiments showed that the proposed model SSHRec performed well when the data set was split using temporal user setting

Table 8.3: Prediction performance of SSHRec with different embedding dimension d

Evaluation Metrics	**K**	**Embedding Size**					
		10	20	30	40	50	100
Precision@K	1	0.1298	0.1725	0.2392	0.2918	0.3009	0.3284
	5	0.0427	0.0740	0.1080	0.1328	0.1395	0.1528
	10	0.0307	0.0538	0.0764	0.0922	0.0977	0.1065
	20	0.0236	0.0411	0.0560	0.0660	0.0701	0.0768
	50	0.0185	0.0301	0.0388	0.0442	0.0468	0.0510
	100	0.0160	0.0242	0.0294	0.0336	0.0349	0.0374
Recall@K	1	0.0968	0.1094	0.1255	0.1364	0.1387	0.1431
	5	0.1107	0.1504	0.1834	0.2034	0.2064	0.2167
	10	0.1224	0.1728	0.2091	0.2302	0.2338	0.2452
	20	0.1395	0.2037	0.2410	0.2637	0.2698	0.2830
	50	0.1845	0.2584	0.3002	0.3246	0.3332	0.3500
	100	0.2348	0.3173	0.3609	0.3922	0.3990	0.4181
MRR		0.1298	0.1725	0.2392	0.2918	0.3009	0.3284
Hitrate@K	1	0.1298	0.1725	0.2392	0.2918	0.3009	0.3284
	5	0.2006	0.3189	0.4256	0.4907	0.5090	0.5347
	10	0.2752	0.4338	0.5486	0.6045	0.6249	0.6510
	20	0.3894	0.5825	0.6896	0.7442	0.7628	0.7899
	50	0.6130	0.8068	0.8695	0.8966	0.9048	0.9173
	100	0.7923	0.9143	0.9356	0.9482	0.9482	0.9512
NDCG@K	1	0.1337	0.1777	0.2464	0.3005	0.3099	0.3382
	5	0.1353	0.1836	0.2377	0.2752	0.2829	0.3030
	10	0.1398	0.1931	0.2461	0.2808	0.2891	0.3081
	20	0.1468	0.2072	0.2607	0.2951	0.3047	0.3243
	50	0.1650	0.2324	0.2892	0.3245	0.3357	0.3570
	100	0.1850	0.2580	0.3165	0.3554	0.3660	0.3886

with training as 80% and test as 20% which indicates that including more historical user interactions better capture users' interest and provide relevant recommendations. Results of the model while using different train and test split strategies are shown in Table 8.4.

8.2 Evaluation of Proposed System with Semantic Enhanced Transition Probability Matrix in Markov Models

In this section, we present the experimental setup along with results and analysis while evaluating our proposed model for semantic and sequential next item recommendation when we used the proposed semantic enhanced transition probability matrix for building the Markov Model.

Table 8.4: Prediction performance of proposed model with different train and test split strategies

	Train and Test Split			
		Leave one Out	Temporal Split	
Evaluation Metric	K	Leave one out	Train=70% Test=30%	Train=80% Test=20%
Precision@K	1	0.3284	0.3805	0.4846
	5	0.1528	0.1906	0.2279
	10	0.1065	0.1290	0.1450
	20	0.0768	0.0868	0.0932
	50	0.0510	0.0537	0.0524
	100	0.0374	0.0375	0.0347
Recall@K	1	0.1431	0.1323	0.1967
	5	0.2167	0.2466	0.3484
	10	0.2452	0.2911	0.3918
	20	0.2830	0.3386	0.4425
	50	0.3500	0.4176	0.5175
	100	0.4181	0.4892	0.5847
NDCG@K	1	0.3382	0.3897	0.4993
	5	0.3030	0.3444	0.4512
	10	0.3081	0.3601	0.4667
	20	0.3243	0.3820	0.4908
	50	0.3570	0.4189	0.5243
	100	0.3886	0.4498	0.5515

8.2.1 Datasets and Implementation Details

Amazon[5] : This dataset includes reviews (ratings, text, helpfulness votes, timestamps), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). The statistics for data set are shown in Table 8.5.

To test our model, we selected the review-K core (which is a subset of the data set where all items have greater than K reviews, where K=5) and product metadata for five categories including auto, baby, garden, office and videos.

Following previous works (He and McAuley, 2016, Rendle et al., 2010) the explicit feedback is converted to implicit feedback by setting a rating score of '1' for a user item

[5]http://jmcauley.ucsd.edu/data/amazon/

Table 8.5: Data set statistics

	Amazon				
Dataset	**No. of Users**	**No. of Items**	**Total Interactions**	**Avg. Interaction per user**	**Avg. Interaction per Item**
Auto	122,492	28,473	369,525	3.02	12.98
Baby	20,434	8,293	169,153	8.28	20.4
Garden	5,376	5,098	59,634	11.09	11.69
Office	7,416	5,490	52,175	9.73	13.15
Video	176,404	19,421	630,513	3.57	32.47

interaction and '0' otherwise.

We implemented the proposed model using python. To compare our proposed model with the baselines, the code from their respective authors (He and McAuley, 2016, Kabbur et al., 2013, Kang and McAuley, 2018b, Rendle et al., 2010) was used to maintain their models' accuracy.

8.2.2 Pre-processing and Hyper-parameter Tuning

For data set partitioning, we adopted commonly used strategies of (i) leave one out (the most recent, i.e., last sequence of each user is used for testing and all remaining sequences for training) and (ii) temporal user splitting (where a percentage of the last interactions of each user is reserved for testing rather than just one). Availability of a rating or review (Amazon) is considered as user-item interaction and we used timestamps to determine the sequential order of actions. Purchases made by each customer were grouped into sequences according to the timestamp. Data was pre-processed to create train and test data. For leave-one-out, the training data was created from those purchase sequences and the last purchase sequence of each customer was used to create the test set for evaluating model's performance. In the temporal user splitting, we used train and test splits of (a) 70%, 30% and (b) 80%, 20%.

Users with purchasing records greater than five are selected. Furthermore, to evaluate the impact of proposed model (SEMMRec) performance on handling sparse data, two variants of each dataset (auto, baby and garden) were created. For example, the data set Auto1 refers to data set which have minimum user interaction as seven and the dataset

Auto2 represents dataset with minimum user interaction as ten. Performance comparison is shown in Fig 8.4 Performance Comparison of proposed Model using Sparse datasets (Two variants of Each Data set).

All the models have some parameters to tune. We follow the reported optimal parameter settings for the baseline methods. On all datasets, for BPR-MF, FISM, FPMC, and Fossil, following settings were adopted: size of latent dimension =100, learning rate= 0.02 and the number of recommendation items = 30. The values of coefficients while computing score measure for products was set to $\alpha = 0.8$ and $\beta = 0.2$ after performing various experiments. For our model, products' embedding dimension d is determined by grid search in the range $\{10, 20, 30, 40, 50, 100\}$, Markov chain based on different order of L (where $L \in \{1, 2, 3, 4, 5\}$), number of Top recommendation items K (where K in $\{1, 5, 10, 20, 30, 50, 100\}$). The values of coefficients (alpha and beta) while computing score measure for products were explored through grid search. Optimal performance was with d=100, $\alpha = 0.8$ and $\beta = 0.2$.

8.2.3 Evaluation Metrices

The model was evaluated on commonly used metrics including Recall@K and NDCG@K as used in (Kang and McAuley, 2018b, Tang and Wang, 2018). The performance was measured at K=30. For a given length of user profile in a test sequence, we predict a list of Top-K items denoted as $\hat{R}_{1:K}$ and the remaining part of the test sequence i.e., ground truth denoted as R. The performance was measured at K=30. The different evaluation metrics are defined as:

- **Recall@K:** Recall is defined as the proportion of relevant items found in the top-K recommendations.
$$\text{Recall@K} = \frac{\left|R \cap \hat{R}_{1:K}\right|}{|R|} \tag{8.5}$$

- **Normalized Discounted Cumulative Gain (NDCG@K):** Evaluates ranking performance by taking the positions of correct items into consideration. NDCG@k is normalized to [0, 1] and a perfect ranking is represented by 1.For each user, the NDCG is computed using the following,

$$\text{NDCG@ K} = \frac{\sum_{k=1}^{K} \frac{I_{c_j}^{k}}{\log_2(k+1)}}{\sum_{k=1}^{m_{c_j}^{K}} \frac{1}{\log_2(k+1)}} \tag{8.6}$$

where $I_{u_i}^{k}$ will be 1 if the kth recommendation for customer c_j is relevant, or it exits in the actual response, and 0 otherwise. Besides, $m_{c_j}^{K}$ is the number of relevant items for customer c_j up to the Kth recommendation. We used the average of NDCG over all users as the final metric of a method. For each user in a test sequence, we predict lists of top-K personalized items where K is in $\{1, 5, 10, 20, 30, 50, 100\}$. We first computed the per-user score for each K and then reported the global average score for all users for each K.

8.2.4 Complexity Analysis

SEMMRec predicts the semantic and sequential product recommendation for the customers. In terms of time complexity, once the transition probability matrix is built, it is trivial to walk through the products (states). As a result, with the number of customers, the model scales well and provides them personalized recommendations.

8.2.5 Baseline Methods for Comparison

To show the effectiveness of our model, we compared the performance of our proposed model SEMMRec with the following advanced existing approaches (Table 8.6) shows the performance comparison for each model:

1. **Baysian Personalized Ranking (BPR-MF).** A state of the art method for non-sequential item recommendation on implicit feedback, utilizing Matrix Factorization Model.
2. **The Factored Item Similarity Models (FISM).** Based on one of the latest recommendation algorithms to capture the relationship between items for personalized recommendations.

Table 8.6: Performance Comparison of proposed Model (SEMMRec) with the baselines

	Datasets											
	Auto		Baby		Garden		Office		Video		Average	
Models \Metric	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
BPR-MF	0.038	0.016	0.058	0.032	0.044	0.021	0.037	0.017	0.043	0.028	0.044	0.022
FISM	0.082	0.047	0.073	0.041	0.085	0.046	0.075	0.049	0.043	0.082	0.071	0.053
FPMC	0.025	0.013	0.047	0.027	0.043	0.024	0.043	0.023	0.030	0.031	0.037	0.023
Fossil	0.081	0.04	0.061	0.034	0.081	0.045	0.082	0.042	0.043	0.067	0.069	0.045
SAS	0.085	0.030	0.067	0.022	0.125	0.043	0.132	0.046	0.186	0.066	0.119	0.041
GRU4Rec	0.0961	0.0331	0.0911	0.0291	0.1011	0.0341	0.1012	0.0341	0.0918	0.0421	0.0962	0.0345
Caser	0.0991	0.0413	0.0971	0.0421	0.0982	0.0441	0.0989	0.0411	0.1024	0.0412	0.0991	0.0419
BERT4Rec	0.1211	0.0710	0.1241	0.0734	0.1521	0.0867	0.1193	0.0973	0.1393	0.1225	0.1311	0.0901
SEMMRec	0.1070	0.0673	0.1254	0.0721	0.1313	0.0731	0.106	0.0602	0.1124	0.1140	0.1164	0.0773

3. **Factorized Personalized Markov Chain (FPMC).** A hybrid approach that combines matrix Factorization (MF) which factorizes the matrix on user-item preferences for learning users' general taste and Markov Chains (MC) that models sequential behavior through a transition graph built over items which predicts users' next action based on the recent actions. However, since this method does not take into account the high-order Markov chain, including it helps in comparative analysis while analysing the effectiveness of high order Markov Chain.

4. **Factorized Sequential Prediction with Item Similarity (Fossil).** A model for sequential recommendations inspired from FPMC and FISM. To increase the performance, it emphasizes the sequential features by combining user preference with high order Markov Chain in a similarity model.

5. **Self Attentive Sequential (SASRec).** It captures long term user preferences by using attention mechanism and makes its predictions based on relatively few actions. The introduction of adaptive self-attention mechanism method models high-order sequence, and it shows high performance for sequential recommendations.

   GRU4Rec. To model sequential dependencies and making predictions in session based recommendation systems, (Hidasi et al., 2016b) proposed this method based on Recurrent Neural Networks (RNN's).

6. **Convolutional Sequence Embeddings (Caser)**. A convolutional neural network (CNN) based method which takes the embedding matrix of the L most recent items and applies convolution operations on it to achieve sequential recommendation (Tang and Wang, 2018).

7. **Bert4Rec.** A Sequential Recommendation model with Bidirectional Encoder Representations from

   Transformer by (Sun et al., 2019b). The model utilizes deep bidirectional self-attention mechanism for modeling user behavioral sequences and learns a bidirectional representation model which makes recommendations by allowing each item in users' historical behavior to integrate information from both left and right sides.

8.2.6 Results and Analysis

Our proposed model SEMMRec gave improved performance in comparison to the baselines on all K tested after incorporating products' meta data to learn product semantics and using semantic similarity measures to compute relationship between products and then utilizing this semantic knowledge while building the transition probability matrix for Markov Model. The aforementioned baseline methods and our method used the same dimension (d =100) to evaluate performance via Recall and NDCG for a uniform comparison. The comparative evaluation results for the given datasets are shown in Table 8.6.

A graphical representation of Table 8.6 is also shown in Fig. 8.2 and it can be seen that BPR-MF and FISM are the recommender systems that only consider user preferences. However, BPR-MF uses a method of factoring a user-item interaction matrix, and FISM factors an item-item similarity matrix. The comparison results show that FISM better reflects user preferences by highlighting the relationships between items. More specifically, FISM shows that on all datasets, the average of Recall is 0.071 and NDCG is 0.053 which is higher than BPR-MF showing that factoring the item-item similarity matrix is technically better.

Fossil and FPMC present the user preference and the sequential patterns for the recommender system. However, Fossil highlights the sequential pattern better than FPMC, and introduces the concept of FISM. In this regard, as illustrated in Table 8.6, Fossil outperforms FPMC on all the datasets; the average of Recall is 0.069 and NDCG 0.045 indicating that learning sequential information about customers' behaviour is important to capture user's long and short term preferences and to improve quality of recommendations and show that the high-order Markov chain and item similarity method are useful for a recommender system in a sequential environment.

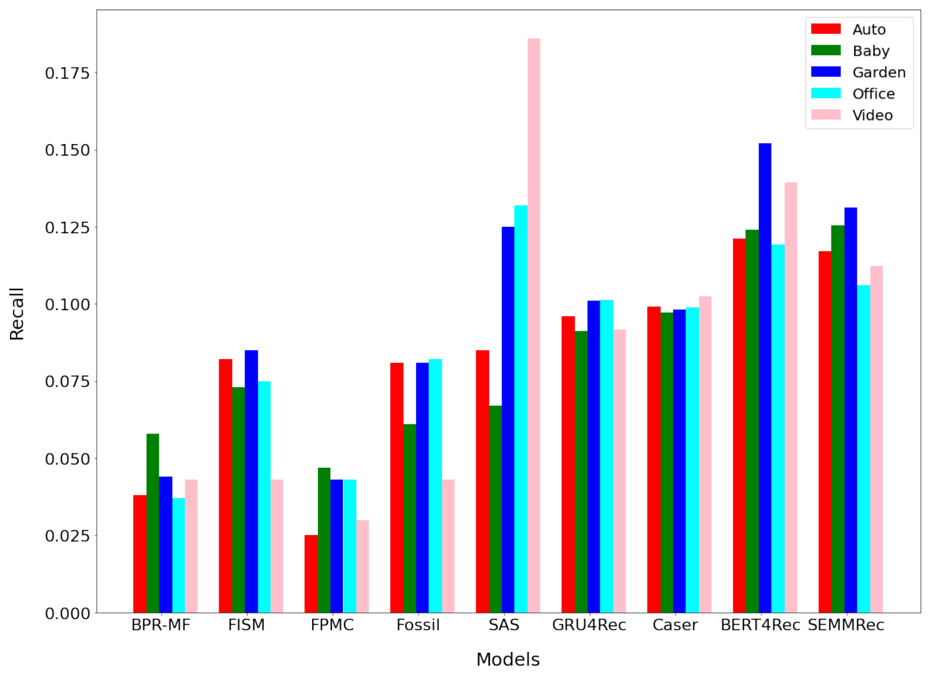
SAS deal with sequential information to implement a sparse sequential recommendation models using deep learning-based approaches. On all datasets, SAS has an average Recall of 0.119 and NDCG of 0.041 which imply that deep learning-based models are useful for sparse sequential recommendation. The comparison between the deep learning-based recommendation methods and conventional recommendation algorithms along with SEMMRec shows that deep learning-based outperforms almost all conventional recommendation algorithms.

However, they require more hyper-parameter tuning and training the model takes much longer. Fig. 8.2 and Table 8.6 illustrate the NDCG and Recall evaluations of the five datasets for all the methods used in the experiment. According to these results, SEMMRec has an improved average Recall of 0.019 and NDCG of 0.019 compared to FISM, an improved average Recall of 0.063 and NDCG of 0027 in comparison to Fossil (that is considered the best performed model among traditional recommendation algorithms) and an improved average Recall of 0.013 and 0.031 as compared to SAS (deep learning based approach).

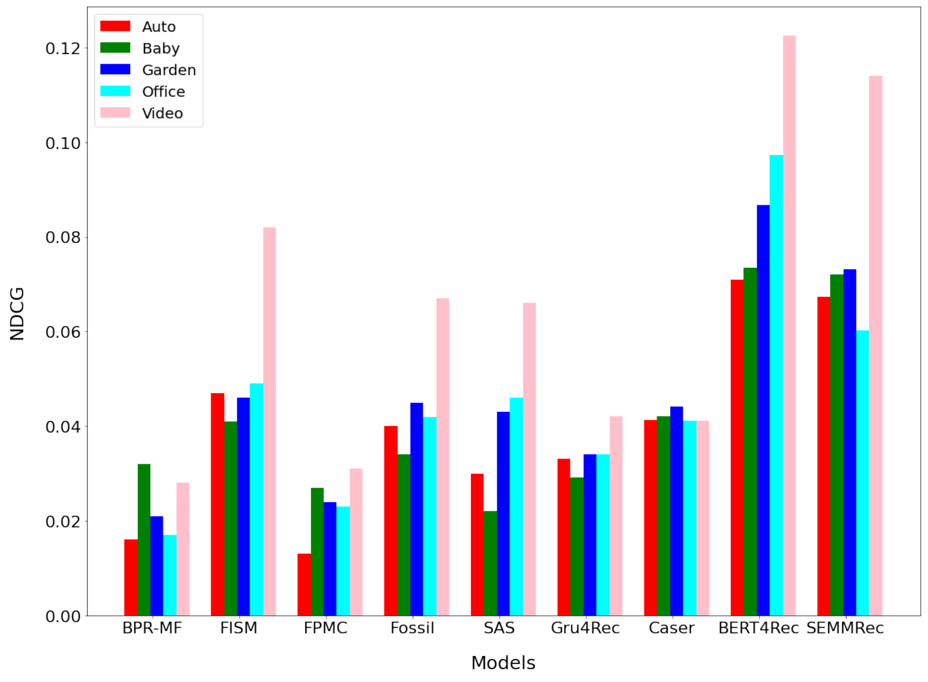
Given performance of all existing methods, it can be said that SEMMRec mostly has better recommendation performance than other recommendation models as it extract semantic knowledge of items from items' meta data (title, description and brand) and customers' purchase histories (co-purchased and co-reviewed products), to compute semantic similarities between items and enriches the sequential next item prediction process of Markov Model by incorporating the semantic knowledge (semantic similarity) into the transition probability matrix thereby generating personalized recommendations.

Furthermore, the results of the proposed model (SEMMRec) on five datasets in Amazon data (Auto, Baby, Office, Garden, Video) are presented in Table 8.6. Here, we report results on all evaluation metrics at a cutoff of K=30. We notice that SEMMRec performed considerably well on all data sets. Next, we will discuss the impact of different order of Markov Chain L), effect of sparsity through our proposed SEMMRec.

8.2.6.0.1 Influence of different Order L of Markov Chain - Next, we analyse the change in performance of the high-order Markov chain based on different order of L where ($L \in \{1, 2, 3, 4, 5\}$). In other words, we compared the performance of recommendations with different values for L. Performance comparisons were made through the 1-order Markov chain. Table 8.7 and Fig. 8.3 shows the performance of our proposed SEMMRec with different values for L in all the datasets. An increase in the number of L elicits an increase in the recommendation performance for most datasets (Baby, Office, Garden, and Video). It means that a high-order Markov chain works well for sequential recommendation in our proposed method.



(A) Recall



(B) NDCG

FIGURE 8.2: Performance comparison of proposed SEMMRec with baseline models on Five Datasets on the basis of (a) Recall and (b) NDCG

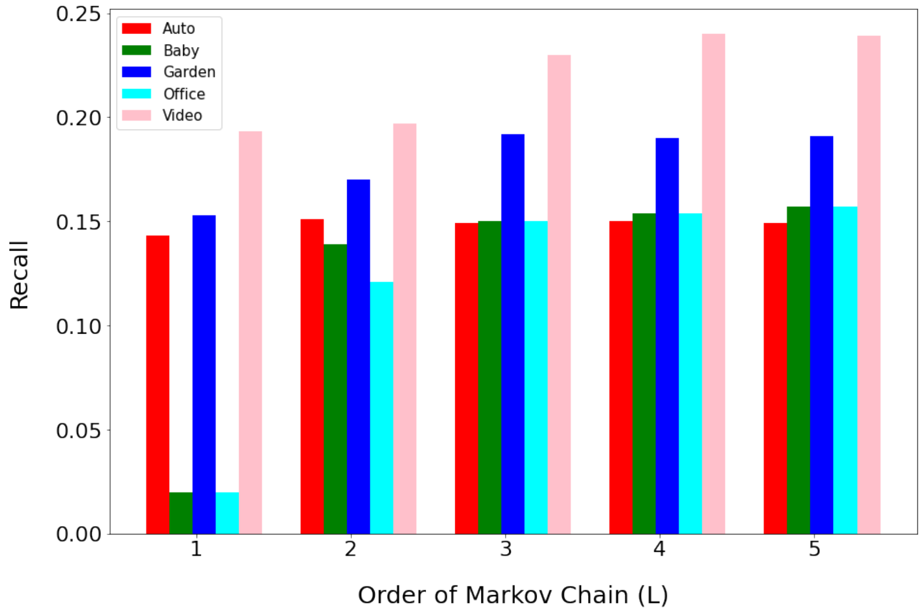
Table 8.7: Effect of different order of L in Markov Chain in SEMMRec

	Metric	**Order of L**				
Data Set		**1**	**2**	**3**	**4**	**5**
Auto	Recall	0.143	0.151	0.149	0.150	0.149
	NDCG	0.089	0.093	0.92	0.092	0.092
Baby	Recall	0.02	0.139	0.15	0.154	0.157
	NDCG	0.092	0.115	0.135	0.140	0.145
Garden	Recall	0.153	0.17	0.192	0.19	0.191
	NDCG	0.093	0.108	0.119	0.12	0.122
Office	Recall	0.02	0.121	0.15	0.154	0.157
	NDCG	0.07	0.083	0.096	0.104	0.104
Video	Recall	0.193	0.197	0.23	0.21	0.20
	NDCG	0.115	0.122	0.124	0.126	0.127

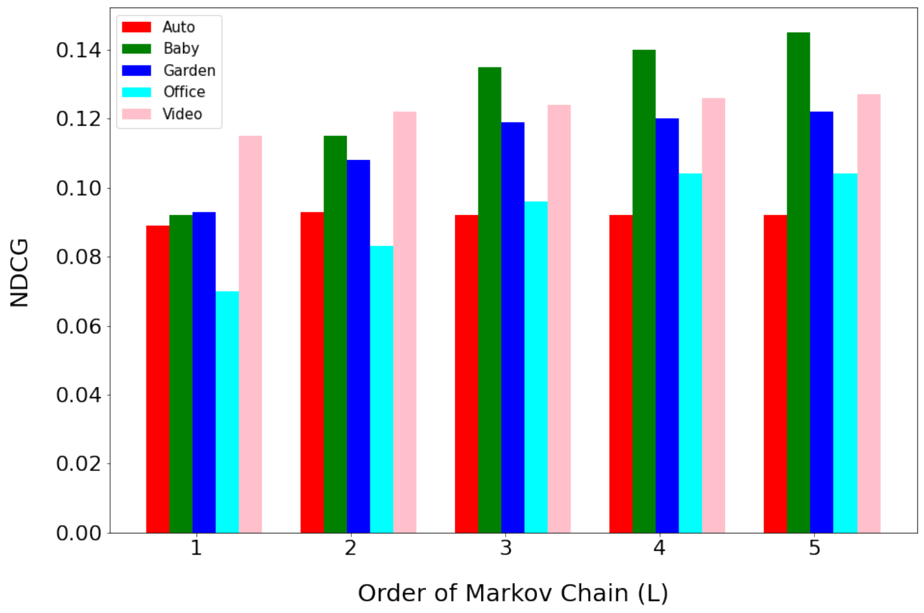
8.2.6.0.2 Handling Sparsity - To show how effectively the proposed SEMMRec deal with sparsity, we generated two variants of each of the Amazon data sets in categories auto, baby and garden. For example, the data set Auto1 refers to data set which have minimum user interaction as seven and the dataset Auto2 represents dataset with minimum user interaction as ten.

Variants for baby and garden data set were also created in the same way. Models' performance was evaluated using same parameters as explained in Sect. 8.2.2. The results are presented in Fig. 8.4 which shows that SEMMRec outperforms the other approaches on different sizes and the length of sequences of datasets. Especially, in small size data set and high-order sequential datasets (e.g. Auto2 and Garden2), SEMMRec and Fossil outperform SAS which is a deep learning-based algorithm. This shows that the conventional machine learning approach compared to deep learning-based approaches appear to have a better performance on the small and high-order sequential dataset. Thus, it can be concluded that our proposed approach is mainly stable on various sparse datasets (with different size and sequence).

8.2.6.0.3 Influence of Train and Test Split - For data set partitioning, we adopted the strategies of (i) leave one out (the most recent, i.e., last sequence of each user is used for testing and all remaining sequences for training) and (ii) temporal user splitting (where a percentage of the last interactions of each user is reserved for testing rather than just one).



(A) Recall



(B) NDCG

FIGURE 8.3: Effects of Different order of L on proposed Model (SEMMRec) Performance based on (a) Recall and (b) NDCG

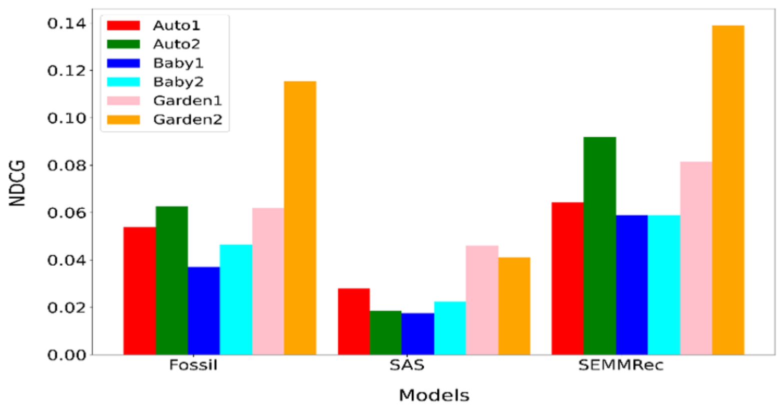


FIGURE 8.4: Performance comparison of proposed model using sparse datasets (two variants of each data set)

Availability of a rating (Amazon) is considered as user-item interaction and we used timestamps to determine the sequential order of actions. Data was pre-processed to create train and test data. Purchases made by each customer were grouped into sequences according to the timestamp. For leave-one-out, the training data was created from those purchase sequences and the last purchase sequence of each customer was used to create the test set for evaluating model's performance. In the temporal user splitting, we used train and test splits of (a) 70%, 30% and (b) 80%, 20%. Users with greater than 5 purchasing records are selected. The experiments showed that the proposed model SEMMRec performed well when the data set was split using temporal user setting with training as 80% and test as 20% which indicates that extracting semantic knowledge of items from items' meta data (title, description and brand) and customers' purchase histories (co-purchased and co-reviewed products) and utilizing it to compute semantic similarities between items and then integrating the semantic knowledge (semantic similarity) into the transition probability matrix enhanced the recommendation process by generating personalized recommendations.

Chapter 9

Conclusion, Limitations and Future Research Directions

This chapter presents a summary of this thesis, its limitations and some future research directions.

9.1 Conclusion

This thesis, proposed a model to explore the effectiveness of utilizing semantic knowledge (meaningful relationships between items) extracted from items' meta data (title, description and brand) and customers' purchase histories to compute semantic similarities between items according to their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products based on their characteristics. The extracted semantic knowledge is then integrated into different phases of recommendation process such as (i) pre-processing, to learn relationships between items, (ii) candidate generation, such as while (a) mining semantic rich sequential patterns, (b) enriching the item matrix in Collaborative Filtering to select Top-N candidates that show semantic and sequential relationships between items and (c) enhancing the Transition Probability Matrix in the Markov Model method and (iii) generate semantic rich and sequential next item recommendations either by using semantic rich (a) Sequential Pattern Rules, (b) item based

Collaborative Filtering or (c) Markov Model depending upon the method used to generate candidate items. Thus, the inclusion of semantic knowledge into all phases of recommendation process can address the issues of sparsity, coldstart, content overspecialization and provide recommendations which are diverse, similar in context and better reflect user's long and short term interests. Experimental results on publicly available e-commerce data sets such as Amazon and Online Retail has shown that the proposed model has improved performance over existing systems. Therefore, the signicant contributions made by this research in context of the research questions described in Chapter 3 and are repeated below.

Research Question 1: How incorporating data from multiple sources can contribute towards personalized recommendations?

Answer: This thesis proposes a comprehensive model which inputs customers' purchase history and products' meta data (e.g., title, description and brand) and extract products' sequential and semantic knowledge according to their (i) usage (e.g., products co-purchased or co-reviewed) and (ii) textual features by finding similarity between products based on their characteristics using distributional hypothesis methods (Prod2Vec, Glove, Doc2vec and TF-IDF) which consider the context of items' usage. Products' semantic feature vectors were learnt by training various distributional models and obtaining products' representations by: (a) Individually training Prod2vec (Grbovic et al., 2015) and Glove (Pennington et al., 2014) models using products from customers' purchase sequences sorted according to the time stamp, (b) utilizing product feature vectors (embeddings) obtained from prod2vec (Grbovic et al., 2015) and Glove (Pennington et al., 2014) to create hybrid embeddings for product representation, (c) individually training TF-IDF (Salton, 1988) and Doc2Vec (Mikolov et al., 2013) models using more product features such as title, description and brand from products' meta data and customers' purchase sequences, where a document represents the collection of products' title, description and brand purchased by the customer sorted according to the time stamp, (d) utilizing product features (embeddings) obtained from TF-IDF (Salton, 1988) and Doc2Vec (Mikolov et al., 2013) to create hybrid embeddings for product representation (Sect. 5.1, Algorithm 2).

The extracted products' semantic knowledge is then utilized in the candidate generation and recommendation phases to generate candidate items that are similar in semantics with target user's preference to generate recommendations that are of interest to user. The items' semantic knowledge is integrated in the candidate generation and recommendation process of sequential recommendation to (i) mine frequent sequential purchase patterns (Sect. 6.1, Algorithm 3), (i) enrich item to item similarity matrix of Collaborative Filtering method and (ii) enhance sequential next item prediction process of first order Markov Model by incorporating the semantic knowledge (semantic similarity) into the Transition Probability Matrix (Sect. 6.1, Algorithm 4) which provides recommendations tailored towards customers' interest.

Research Question 2: Can items' semantic knowledge be integrated to enhance the performance of item based Collaborative Filtering Systems?

Answer: Yes. This is obtained by computing cosine similarity between the products' semantic feature vectors obtained from customers' purchase histories and products' meta data. This semantic similarity between items is then used to create an item-item semantic similarity matrix for item based Collaborative Filtering (Chapter 6). To further enhance the item based Collaborative Filtering with items semantic and sequential relationships, a weighted products' score measure is proposed which is computed through semantic similarity, confidence and lift measures between a pair of products. The item similarity matrix is then populated with this score measure where each entry in the matrix represents semantic and sequential relationship (in terms of a score value) between a pair of products (Sect. 7.1, Algorithm 6). The matrix can then be used to generate semantic and sequential next item recommendation by taking each item in target user's profile and retrieving a list of Top-K items with a high score.

Research Question 3: How does incorporating semantics to sequential pattern mining methods address their limitations and enhance the accuracy of recommendations?

Answer: This is accomplished by integrating semantic knowledge extracted from customers' purchase histories and products' meta data to enhance the mining process. The first step is to determine Top-N customers based on semantic similarities between products in their purchase sequences, where each purchase sequence will be represented as an aggregate vector of all products in the sequence (Section 6.1.3). This is analogous to representing a sentence as collection of words and then creating a database of these semantically similar purchase sequences to mine frequent semantic and sequential patterns using PrefixSpan

(Jian Pei et al., 2001). During the mining process, the semantic information of products from the item similarity matrix along with their support count is used to prune patterns of products that are below the specified semantic similarity threshold and minimum support threshold (Section 6.1, Algorithm 3). Later, in the recommendation phase, the extracted semantic rich sequential patterns are used to generate rules to recommend items that are similar in semantics and can be purchased in sequential order (Algorithm 5).

Research Question 4: Can the inclusion of semantic information solve the problems in the recommendation systems and how?

Answer: To address the items' cold start issue, this thesis proposes to compute semantic and sequential relationships between items from customers' purchase history and products' meta data (e.g., title, description and brand) and extract products' semantic knowledge according to their (i) usage (e.g., products co-purchased or co-reviewed) and (ii) textual features by finding similarity between products based on their characteristics. These relationships obtained in the form of product feature vector representations can then be used to compute similarities between existing and a new item to guarantee the appearance of this newly introduced item into the recommendation set without the use of any explicit ratings.

For a new user issue, where no previous history exists, the current user behaviour (e.g., what items she is browsing), are considered to find items that are similar in semantics. Then, for recommendation using user-based CF, Top-N neighbors (customers' whose interest is similar to the target customer) are identified based on the products' in their purchase sequence. This is based on semantic similarities between products in their purchase sequences, where each purchase sequence will be represented as an aggregate vector of all products in the sequence (Section 6.1.3) and each user will be represented as an aggregate vector of her purchase sequences to compute Top-N neighbors. This is analogous to representing a sentence as collection of words. *Sparsity* (lack of user item interactions) is addressed as an item's relationship with other items is computed on the basis of item's textual features and usage context (e.g., items co-purchased, co-viewed). Hence, to compute similarities between items, no explicit information (ratings) are required. The huge search space issue is also addressed by narrowing down the search space during the mining process by proposing to prune patterns on the basis of semantic similarity threshold in addition to traditional pruning method of using the support count.

Furthermore, a first order Markov Model's Transition Probability Matrix is enhanced with items' semantic knowledge by first (a) creating a product frequency matrix based on the sequential occurrence of each consecutive pair of products using the customers' historical purchases, (b) creating a Transition probability Matrix by normalizing the matrix obtained in step (a) as explained in Sections 6.2.1 and 6.2.2 and then integrating the semantic information of products from the item similarity matrix by computing a weighted score (based on their semantic and sequential occurrence) for each consecutive pair of products in the Transition Probability Matrix (Section 6.2, Algorithm 4). This provides us the items that are similar in semantics and purchased in sequential order. A personalization vector is then computed for each customer based on any product in its most recent purchases (as a starting probability vector for the customer to begin at state k) in order to determine the probabilities for landing at state k+1 (i.e., recommending a personalized next item by initiating the customers' journey from state k which represent any of the products purchased by the customer during the last time stamp) and then determine the next product for purchase (state k+1) based on the weighted score (Sect. 7.2, Algorithm 7).

9.2 Limitations

1. **Adaptability for recommendation in other domains** The proposed system has been evaluated and tested for e-commerce products as according to the thesis scope, however, it still needs to be investigated whether the proposed approach can also be adapted to generate recommendations in other domains such as news recommendation, music recommendation to name a few.

2. **Availability of dataset from other sources** Besides click stream and purchase data, there is a lack of publicly available e-commerce data set that include various other aspects of user interactions such as items added to their wish list, added to cart but not purchased. Therefore, the proposed model could not be evaluated based on these user-item interactions to interpret the impact of these data sources while generating the semantic and sequential relationships between items.

9.3 Future Research Directions

Research on Sequential Recommendation Systems has gained attention in the past several years. While summarizing and categorizing the various research directions followed, we ascertain further open research directions including:

1. **Context-adaptation in Sequential Recommendation Systems**

   Estimating the current context for a user to understand its preferences can greatly impact the quality of recommendation and therefore increase user satisfaction. For example, a context may represents a set of factors or situations under which a user interacted with an item such as time, location, surroundings, purpose of purchase, device and occasion while interacting with an item. Knowing the context in sequential recommender systems is more essential as user's intent are short term and may evolve quickly with time. For example, in an e-commerce platform, it is important to determine if the user is going over the catalog to find different range of options or she is interested in just reviewing shortlisted candidate items for purchase. However, most existing Sequential Recommendation Systems ignore this significant aspect. Hence, further work in this direction can be promising.

2. **Social influence in Sequential Recommendation Systems**

   Users tend to trust more on the recommendations from their friends (social network) than recommendations by unknown people. Collecting social information of users such as facebook friends, followers on twitter, gathering preferences of their friends from social networks and then utilize their ratings on items to generate recommendations for the target user needs to be taken into account in Sequential Recommendation Systems. A common example could be recommending movie to watch based on the movies recently watched by target user's friends or recommending products by collecting information on products bought by target users' friends and are similar to her interest.

3. **Cross-domain Sequential Recommendation Systems**

   Cross domain recommendation aims to leverage data in different domains by transferring knowledge from source domain to target domain (for example, recommending books on the basis of movies watched by the user), hence amplifying the target domains' recommendation performance. Future research can investigate deep learning approaches to determine the characteristics of sequential data in source and target domains and design Sequential Recommendation Systems generating cross-domain recommendations.

4. **Explainability in Sequential Recommendation Systems**

   Recently, most advanced neural network and deep learning based models are deemed as a black box and lack explainability for users and model practitioners. Therefore, designing Sequential Recommendation Systems which are explainable is significant as without knowing the reasons behind recommendations, users' may not trust these and hence do not take an action accordingly (e.g., purchasing an item). Similarly, it is vital for the practitioners to understand the influence of various factors such as data, features, and other model hyper-parameters on the model output (i.e., recommendations).

5. **Incorporating general trends and additional data**

   Usually, few specific types of user interactions (e.g., items views, clicks and purchases) are considered while creating users' profile (recording its preferences). However, in real world, other rich information sources are available relevant to items (e.g., add to favourites, add to wish-list, add to cart, trending items) and users (e.g., navigation across different categories, purchase behaviours during several occasions). Focus on these additional information sources can also impact the quality of recommendations.

6. **Comprehensive and standardised evaluations across different Models**

   There has been a debate that only complex and advanced deep learning models cannot always guarantee better and more robust recommender systems. Additionally, one critical issue for evaluation in Sequential Recommendation Systems is the lack of effective standardised benchmarks. Therefore, it is imperative to lay emphasis on benchmarking study for standardised evaluations.

7. **Domain specific sequential recommendation**

   Most of the research conducted for sequential recommendation is considered to be applicable to all domains (e.g., music, news, movies, product recommendation). However, to address the needs for real world applications, more focus can be given to design recommendation algorithms for a particular domain based on domain specific features and hence have common data sets and standard baselines for comparison. For example, in case of movie recommendation, common features include “genre”, “artists”, “director”, “writer”,“release year“, “awards” etc., to being with, however, in case of e-commerce domain for product recommendation, product features vary not just in comparison to other domains but also between various product categories as most of the product information (features) is embedded in the text descriptions. For example, consider the description of a baby girl clothing with description as “ Cute Blue Red Cranberry Taped Girl Ruffle long Sleeve Sunflower Dress”, so here “Cranberry” and “Sunflower” may refer to a “flavour” , an “ingredient’ or simply a “color”. Accurate identification of these features is important to generate accurate and relevant recommendations.

8. **Inclusion of products’ semantic knowledge**

   Extracting products’ semantic knowledge (e.g., using textual features and context) and then including it in the sequential recommendation process can improve the recommendation process. Nasir et al (Nasir and Ezeife, 2020, Nasir et al., 2021) proposed to use customers’ purchase history and products’ meta data (e.g., title, description and brand) and then extract products’ sequential and semantic knowledge according to their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products based on their characteristics considering the context of items’ usage.

9. **Exploring consequential bond between customers’ different sequential interactions**

   Consequential bond (similarity between click and purchase sequences) (Bhatta et al., 2019) is useful to find similarities between customers’ click and purchase sequences. This relationship (consequential bond) can be further extended to other user-item interactions including items viewed, added to cart, add to wish list to further explore sequential dependencies between users’ various interactions and hence recommend items of interest.

10. **Mining historical (long term) and short term user preferences**

    Considering the shift in users' interest and preferences, it will be good to consider users' historical (long term) records (e.g., past purchases) along with their current interests (e.g., short term user–item interactions). This can facilitate in learning their static (long term) preferences (e.g., a particular clothing brand/style) and dynamic short term intent (e.g., a particular colour). Models integrating these can lead to improved recommendations tailored to customers' needs.

Bibliography

Adomavicius, G. and Tuzhilin, A. Context-Aware Recommender Systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 217–253. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_7. URL `https://doi.org/10.1007/978-0-387-85820-3_7`.

Aggarwal, C. C. An Introduction to Recommender Systems. In Aggarwal, C. C., editor, *Recommender Systems: The Textbook*, pages 1–28. Springer International Publishing, Cham, 2016. ISBN 978-3-319-29659-3. doi: 10.1007/978-3-319-29659-3_1. URL `https://doi.org/10.1007/978-3-319-29659-3_1`.

Agrawal, R. and Srikant, R. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, March 1995. doi: 10.1109/ICDE.1995.380415.

Agrawal, R., Srikant, R., Road, H., and Jose, S. Fast Algorithms for Mining Association Rules. page 32.

Alkilany, A. A. A. An overview: temporal-side of sequential patterns discovery. *International Journal of Data Mining & Knowledge Management Process*, 3(1):1, 2013. Publisher: Academy & Industry Research Collaboration Center (AIRCC).

Asher, R. E. and Simpson, J. M. Y. *The Encyclopedia of Language and Linguistics*. Pergamon, 1993.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. Sequential PAttern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 429–435, New York, NY, USA, July 2002. Association for Computing Machinery. ISBN 978-1-58113-567-1. doi: 10.1145/775047.775109. URL `https://doi.org/10.1145/775047.775109`.

Bai, T., Nie, J.-Y., Zhao, W. X., Zhu, Y., Du, P., and Wen, J.-R. An attribute-aware neural attentive model for next basket recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1201–1204, 2018.

Basile, P., de Gemmis, M., Gentile, A. L., Lops, P., and Semeraro, G. UNIBA: JIGSAW algorithm for Word Sense Disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 398–401, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S07-1088`.

Bernhard, S. D., Leung, C. K., Reimer, V. J., and Westlake, J. Clickstream Prediction Using Sequential Stream Mining Techniques with Markov Chains. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, IDEAS '16, pages 24–33, New York, NY, USA, July 2016. Association for Computing Machinery. ISBN 978-1-4503-4118-9. doi: 10.1145/2938503.2938535. URL `https://doi.org/10.1145/2938503.2938535`.

Bhatta, R., Ezeife, C. I., and Butt, M. N. Mining Sequential Patterns of Historical Purchases for E-commerce Recommendation. In Ordonez, C., Song, I.-Y., Anderst-Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 57–72, Cham, 2019. Springer International Publishing. ISBN 978-3-030-27520-4. doi: 10.1007/978-3-030-27520-4_5.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, July 2013. ISSN 0950-7051. doi: 10.1016/j.knosys.2013.03.012. URL `http://www.sciencedirect.com/science/article/pii/S0950705113001044`.

Brafman, R. I., Heckerman, D., and Shani, G. Recommendation as a Stochastic Sequential Decision Problem. page 10.

Burke, R. Hybrid Web Recommender Systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture

Notes in Computer Science, pages 377–408. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9. doi: 10.1007/978-3-540-72079-9_12. URL https://doi.org/10.1007/978-3-540-72079-9_12.

Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R., and García-Sánchez, F. Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications*, 39(12):10990–11000, September 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.03.025. URL http://www.sciencedirect.com/science/article/pii/S0957417412004952.

Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., and Zha, H. Sequential Recommendation with User Memory Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 108–116, Marina Del Rey CA USA, February 2018. ACM. ISBN 978-1-4503-5581-0. doi: 10.1145/3159652.3159668. URL https://dl.acm.org/doi/10.1145/3159652.3159668.

Cheng, W., Yin, G., Dong, Y., Dong, H., and Zhang, W. Collaborative Filtering Recommendation on Users' Interest Sequences. *PLOS ONE*, 11(5):e0155739, May 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0155739. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155739. Publisher: Public Library of Science.

Choi, K., Yoo, D., Kim, G., and Suh, Y. A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications*, 11(4):309–317, July 2012. ISSN 1567-4223. doi: 10.1016/j.elerap.2012.02.004. URL http://www.sciencedirect.com/science/article/pii/S156742231200018X.

Christakopoulou, K., Beutel, A., Li, R., Jain, S., and Chi, E. H. Q&amp;R: A Two-Stage Approach toward Interactive Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 139–148, New York, NY, USA, July 2018. Association for Computing Machinery. ISBN 978-1-4503-5552-0. doi: 10.1145/3219819.3219894. URL https://doi.org/10.1145/3219819.3219894.

de Gemmis, M., Lops, P., Musto, C., Narducci, F., and Semeraro, G. Semantics-Aware Content-Based Recommender Systems. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 119–159. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7637-6. doi: 10.1007/978-1-4899-7637-6_4. URL `https://doi.org/10.1007/978-1-4899-7637-6_4`.

Deshpande, M. and Karypis, G. Selective Markov Models for Predicting Web Page Accesses. *ACM Transactions on Internet Technology*, 4(2):22.

Di Noia, T., Mirizzi, R., Ostuni, V. C., Romito, D., and Zanker, M. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems - I-SEMANTICS '12*, page 1, Graz, Austria, 2012. ACM Press. ISBN 978-1-4503-1112-0. doi: 10.1145/2362499.2362501. URL `http://dl.acm.org/citation.cfm?doid=2362499.2362501`.

Donkers, T., Loepp, B., and Ziegler, J. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 152–160, 2017.

Dziugaite, G. K. and Roy, D. M. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. *Collaborative Filtering Recommender Systems*. Now Publishers Inc, 2011. ISBN 978-1-60198-442-5.

Ezeife, C. I., Lu, Y., and Liu, Y. PLWAP sequential mining: open source code. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, OSDM '05, pages 26–35, New York, NY, USA, August 2005. Association for Computing Machinery. ISBN 978-1-59593-210-5. doi: 10.1145/1133905.1133910. URL `https://doi.org/10.1145/1133905.1133910`.

FIRTH, J. R. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, 1957. URL `https://ci.nii.ac.jp/naid/10020680394/`. Publisher: Basil Blackwell.

Fournier-Viger, P. and Lin, J. C.-W. A Survey of Sequential Pattern Mining. page 24.

Gabrilovich, E. and Markovitch, S. Wikipedia-based Semantic Interpretation for Natural Language Processing. *Journal of Artificial Intelligence Research*, 34:443–498, March 2009.

ISSN 1076-9757. doi: 10.1613/jair.2669. URL https://www.jair.org/index.php/jair/article/view/10595.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. E-commerce in Your Inbox: Product Recommendations at Scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1809–1818, New York, NY, USA, August 2015. Association for Computing Machinery. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788627. URL https://doi.org/10.1145/2783258.2788627.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

Guo, W., Wang, S., Lu, W., Wu, H., Zhang, Q., and Shao, Z. Sequential dependency enhanced graph neural networks for session-based recommendations. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2021.

Gurbanov, T. and Ricci, F. Action prediction models for recommender systems based on collaborative filtering and sequence mining hybridization. In *Proceedings of the Symposium on Applied Computing*, pages 1655–1661, 2017.

Harris, Z. S. Distributional Structure. *WORD*, 10(2-3):146–162, August 1954. ISSN 0043-7956, 2373-5112. doi: 10.1080/00437956.1954.11659520. URL http://www.tandfonline.com/doi/full/10.1080/00437956.1954.11659520.

He, R. and McAuley, J. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200, Barcelona, Spain, December 2016. IEEE. ISBN 978-1-5090-5473-2. doi: 10.1109/ICDM.2016.0030. URL http://ieeexplore.ieee.org/document/7837843/.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS. page 10, 2016a.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv:1511.06939 [cs]*, March 2016b. URL `http://arxiv.org/abs/1511.06939`. arXiv: 1511.06939.

Hidasi, B., Quadrana, M., Karatzoglou, A., and Tikk, D. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 241–248, New York, NY, USA, September 2016c. Association for Computing Machinery. ISBN 978-1-4503-4035-9. doi: 10.1145/2959100.2959167. URL `https://doi.org/10.1145/2959100.2959167`.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Hsu, K.-C., Chou, S.-Y., Yang, Y.-H., and Chi, T.-S. Neural network based next-song recommendation. *arXiv preprint arXiv:1606.07722*, 2016.

Huang, J., Zhao, W. X., Dou, H., Wen, J.-R., and Chang, E. Y. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514, 2018a.

Huang, J., Zhao, W. X., Dou, H., Wen, J.-R., and Chang, E. Y. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514, Ann Arbor MI USA, June 2018b. ACM. ISBN 978-1-4503-5657-2. doi: 10.1145/3209978.3210017. URL `https://dl.acm.org/doi/10.1145/3209978.3210017`.

Hunt, J. W. and MacIlroy, M. D. *An algorithm for differential file comparison.* Bell Laboratories Murray Hill, 1976.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. *Recommender Systems: An Introduction.* Cambridge University Press, September 2010. ISBN 978-1-139-49259-1. Google-Books-ID: eygTJBd_U2cC.

Jian Pei, Jiawei Han, Mortazavi-Asl, B., Pinto, H., Qiming Chen, Dayal, U., and Mei-Chun Hsu. PrefixSpan,: mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th International Conference on Data Engineering*, pages 215–224, Heidelberg, Germany, 2001. IEEE Comput. Soc. ISBN 978-0-7695-1001-9. doi: 10.1109/ICDE.2001.914830. URL `http://ieeexplore.ieee.org/document/914830/`.

Kabbur, S., Ning, X., and Karypis, G. FISM: factored item similarity models for top-N recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667, Chicago Illinois USA, August 2013. ACM. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2487589. URL `https://dl.acm.org/doi/10.1145/2487575.2487589`.

Kang, W.-C. and McAuley, J. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018a.

Kang, W.-C. and McAuley, J. Self-Attentive Sequential Recommendation. *arXiv:1808.09781 [cs]*, August 2018b. URL `http://arxiv.org/abs/1808.09781`. arXiv: 1808.09781.

Kang, W.-C., Wan, M., and McAuley, J. Recommendation through mixtures of heterogeneous item relationships. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1143–1152, 2018.

Kim, Y. S. and Yum, B.-J. Recommender system based on click stream data using association rule mining. *Expert Systems with Applications*, 38(10):13320–13327, September 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.04.154. URL `http://www.sciencedirect.com/science/article/pii/S0957417411006816`.

Kim, Y. S., Yum, B.-J., Song, J., and Kim, S. M. Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Systems with Applications*, 28(2):381–393, February 2005. ISSN 0957-4174. doi: 10.1016/j.eswa.2004.10.017. URL `http://www.sciencedirect.com/science/article/pii/S0957417404001368`.

Koren, Y. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 447–456, New York, NY, USA, June 2009. Association for Computing Machinery.

ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557072. URL `https://doi.org/10.1145/1557019.1557072`.

Le, Q. and Mikolov, T. Distributed Representations of Sentences and Documents. page 9.

Leacock, C. and Chodorow, M. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

Lee, D., Chuang, H., and Seamons, K. Document ranking and the vector-space model. *IEEE Software*, 14(2):67–75, 1997. doi: 10.1109/52.582976.

Li, J., Li, L., Wu, Y., and Chen, S. An Improved Recommender Based on Hidden Markov Model. page 5.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., and Ma, J. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

Li, T., Choi, M., Fu, K., and Lin, L. Music Sequence Prediction with Mixture Hidden Markov Models. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6128–6132, December 2019. doi: 10.1109/BigData47090.2019.9005695.

Lian, D., Xie, X., and Chen, E. Discrete matrix factorization and extension for fast item recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1919–1933, 2019.

Lian, J., Zhang, F., Xie, X., and Sun, G. Cccfnet: a content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the 26th international conference on World Wide Web companion*, pages 817–818, 2017.

Liu, D.-R., Lai, C.-H., and Lee, W.-J. A hybrid of sequential rules and collaborative filtering for product recommendation. *Information Sciences*, 179(20):3505–3519, 2009.

Liu, Q., Zeng, Y., Mokhosi, R., and Zhang, H. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1831–1839, 2018.

Lops, P., de Gemmis, M., and Semeraro, G. Content-based Recommender Systems: State of the Art and Trends. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 73–105. Springer US, Boston, MA, 2011. ISBN 978-0-387-85820-3. doi: 10.1007/978-0-387-85820-3_3. URL `https://doi.org/10.1007/978-0-387-85820-3_3`.

Lowe, W. Towards a Theory of Semantic Space. page 7.

Mabroukeh, N. R. and Ezeife, C. I. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1):3:1–3:41, December 2010. ISSN 0360-0300. doi: 10.1145/1824795.1824798. URL `https://doi.org/10.1145/1824795.1824798`.

Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963773. URL `https://doi.org/10.1145/963770.963773`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

Miller, G. A. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995. ISSN 00010782. doi: 10.1145/219717.219748. URL `http://portal.acm.org/citation.cfm?doid=219717.219748`.

Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. Discovery and evaluation of aggregate usage profiles for web personalization. *Data mining and knowledge discovery*, 6(1):61–82, 2002.

Mooney, C. H. and Roddick, J. F. Sequential pattern mining – approaches and algorithms. *ACM Computing Surveys*, 45(2):1–39, February 2013. ISSN 0360-0300, 1557-7341. doi: 10.1145/2431211.2431218. URL `https://dl.acm.org/doi/10.1145/2431211.2431218`.

Morris, M. R., Teevan, J., and Panovich, K. What do people ask their social networks, and why? a survey study of status message q&amp;a behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1739–1748, New York, NY, USA, April 2010. Association for Computing Machinery. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753587. URL `https://doi.org/10.1145/1753326.1753587`.

Nasir, M. and Ezeife, C. I. Semantics Embedded Sequential Recommendation for E-Commerce Products (SEMSRec). In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 270–274, December 2020. doi: 10.1109/ASONAM49781.2020.9381352. ISSN: 2473-991X.

Nasir, M., Ezeife, C. I., and Gidado, A. Improving e-commerce product recommendation using semantic context and sequential historical purchases. *Social Network Analysis and Mining*, 11(1):82, September 2021. ISSN 1869-5469. doi: 10.1007/s13278-021-00784-6. URL `https://doi.org/10.1007/s13278-021-00784-6`.

Nielsen, M. A. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

Parameswaran, A. G., Koutrika, G., Bercovitz, B., and Garcia-Molina, H. Recsplorer: recommendation algorithms based on precedence mining. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 87–98, 2010.

Pasricha, R. and McAuley, J. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 63–71, 2018.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, November 2004. ISSN 1558-2191. doi: 10.1109/TKDE.2004.77. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Pennington, J., Socher, R., and Manning, C. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `http://aclweb.org/anthology/D14-1162`.

Polyzou, A. and Karypis, G. Scholars Walk: A Markov Chain Framework for Course Recommendation. page 6, 2019.

Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 130–137, New York, NY, USA, August 2017. Association for Computing Machinery. ISBN 978-1-4503-4652-8. doi: 10.1145/3109859.3109896. URL `https://doi.org/10.1145/3109859.3109896`.

Ren, P., Chen, Z., Li, J., Ren, Z., Ma, J., and De Rijke, M. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4806–4813, 2019.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 811–820, New York, NY, USA, April 2010. Association for Computing Machinery. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772773. URL `https://doi.org/10.1145/1772690.1772773`.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. page 10, 2009.

Ricci, F., Rokach, L., and Shapira, B. Recommender Systems: Introduction and Challenges. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 1–34. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7637-6. doi: 10.1007/978-1-4899-7637-6_1. URL `https://doi.org/10.1007/978-1-4899-7637-6_1`.

Rubino, G. and Sericola, B. *Markov Chains and Dependability Theory.* Cambridge University Press, Cambridge, 2014. ISBN 978-1-139-05170-5. doi: 10.1017/CBO9781139051705. URL `http://ebooks.cambridge.org/ref/id/CBO9781139051705`.

Rudin, C., Letham, B., Salleb-Aouissi, A., Kogan, E., and Madigan, D. Sequential event prediction with association rules. In *Proceedings of the 24th annual conference on learning theory*, pages 615–634. JMLR Workshop and Conference Proceedings, 2011.

Sachdeva, N., Gupta, K., and Pudi, V. Attentive neural architecture incorporating song features for music recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 417–421, 2018.

Sachdeva, N., Manco, G., Ritacco, E., and Pudi, V. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 600–608, 2019.

Sahlgren, M. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53, 2008.

Sahoo, N., Singh, P. V., and Mukhopadhyay, T. A hidden markov model for collaborative filtering. *MIS quarterly*, pages 1329–1356, 2012.

Saini, S., Saumya, S., and Singh, J. P. Sequential Purchase Recommendation System for E-Commerce Sites. In Saeed, K., Homenda, W., and Chaki, R., editors, *Computer Information Systems and Industrial Management*, Lecture Notes in Computer Science, pages 366–375, Cham, 2017. Springer International Publishing. ISBN 978-3-319-59105-6. doi: 10.1007/978-3-319-59105-6_31.

Salehi, M. and Nakhai Kamalabadi, I. Hybrid recommendation approach for learning material based on sequential pattern of the accessed material and the learner's preference tree. *Knowledge-Based Systems*, 48:57–69, August 2013. ISSN 0950-7051. doi: 10.1016/j.knosys.2013.04.012. URL `http://www.sciencedirect.com/science/article/pii/S095070511300124X`.

Salton, G. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley series in computer science. Addison-Wesley, Reading, Mass, 1988. ISBN 978-0-201-12227-5.

Sarukkai, R. R. Link prediction and path analysis using Markov chains. *Computer Networks*, 33(1-6):377–386, June 2000. ISSN 13891286. doi: 10.1016/S1389-1286(00)00044-X. URL `https://linkinghub.elsevier.com/retrieve/pii/S138912860000044X`.

Schafer, J. B., Konstan, J. A., and Riedl, J. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*, 5(1):115–153, January 2001. ISSN 1573-756X. doi: 10.1023/A:1009804230409. URL https://doi.org/10.1023/A:1009804230409.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. Collaborative Filtering Recommender Systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science, pages 291–324. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9. doi: 10.1007/978-3-540-72079-9_9. URL https://doi.org/10.1007/978-3-540-72079-9_9.

Semeraro, G., Degemmis, M., Lops, P., and Basile, P. Combining learning and word sense disambiguation for intelligent user profiling. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2856–2861, San Francisco, CA, USA, January 2007. Morgan Kaufmann Publishers Inc.

Semeraro, G., Lops, P., Basile, P., and de Gemmis, M. Knowledge infusion into content-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 301–304, New York, NY, USA, October 2009. Association for Computing Machinery. ISBN 978-1-60558-435-5. doi: 10.1145/1639714.1639773. URL https://doi.org/10.1145/1639714.1639773.

Shani, G., Heckerman, D., and Brafman, R. I. An MDP-Based Recommender System. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005. ISSN ISSN 1533-7928. URL https://www.jmlr.org/papers/v6/shani05a.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Smirnova, E. and Vasile, F. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, DLRS 2017, pages 2–9, New York, NY, USA, August 2017. Association for Computing Machinery. ISBN 978-1-4503-5353-3. doi: 10.1145/3125486.3125488. URL https://doi.org/10.1145/3125486.3125488.

Song, W. and Yang, K. Personalized recommendation based on weighted sequence similarity. In *Practical Applications of Intelligent Systems*, pages 657–666. Springer, 2014.

Su, Q. and Chen, L. A method for discovering clusters of e-commerce interest patterns using click-stream data. *Electronic Commerce Research and Applications*, 14(1):1–13, January 2015. ISSN 1567-4223. doi: 10.1016/j.elerap.2014.10.002. URL `http://www.sciencedirect.com/science/article/pii/S1567422314000726`.

Su, X. and Khoshgoftaar, T. M. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009:1–19, 2009. ISSN 1687-7470, 1687-7489. doi: 10.1155/2009/421425. URL `https://www.hindawi.com/archive/2009/421425/`.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019a.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1441–1450, Beijing China, November 2019b. ACM. ISBN 978-1-4503-6976-3. doi: 10.1145/3357384.3357895. URL `https://dl.acm.org/doi/10.1145/3357384.3357895`.

Tan, Y. K., Xu, X., and Liu, Y. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, pages 17–22, New York, NY, USA, September 2016. Association for Computing Machinery. ISBN 978-1-4503-4795-2. doi: 10.1145/2988450.2988452. URL `https://doi.org/10.1145/2988450.2988452`.

Tang, J. and Wang, K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. *arXiv:1809.07426 [cs]*, September 2018. URL `http://arxiv.org/abs/1809.07426`. arXiv: 1809.07426.

Tang, J., Belletti, F., Jain, S., Chen, M., Beutel, A., Xu, C., and H. Chi, E. Towards neural mixture recommender for long range dependent user sequences. In *The World Wide Web Conference*, pages 1782–1793, 2019.

Tavakol, M. and Brefeld, U. Factored mdps for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 33–40, 2014.

Tuan, T. X. and Phuong, T. M. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 138–146, 2017.

Turney, P. D. and Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188, February 2010. ISSN 1076-9757. doi: 10.1613/jair.2934. URL `https://www.jair.org/index.php/jair/article/view/10640`.

Unger, M., Bar, A., Shapira, B., and Rokach, L. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178, 2016.

Villatel, K., Smirnova, E., Mary, J., and Preux, P. Recurrent Neural Networks for Long and Short-Term Sequential Recommendation. *arXiv:1807.09142 [cs, stat]*, July 2018. URL `http://arxiv.org/abs/1807.09142`. arXiv: 1807.09142.

Wan, M. and McAuley, J. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems*, pages 86–94, 2018.

Wan, S., Lan, Y., Wang, P., Guo, J., Xu, J., and Cheng, X. Next basket recommendation with neural networks. In *RecSys Posters*, 2015.

Wang, C., Niepert, M., and Li, H. Recsys-dan: discriminative adversarial networks for cross-domain recommender systems. *IEEE transactions on neural networks and learning systems*, 31(8):2731–2740, 2019a.

Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 403–412, 2015.

Wang, Q., Yin, H., Hu, Z., Lian, D., Wang, H., and Huang, Z. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2467–2475, 2018a.

Wang, S., Hu, L., Cao, L., Huang, X., Lian, D., and Liu, W. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018b.

Wang, S., Hu, L., Wang, Y., Sheng, Q. Z., Orgun, M., and Cao, L. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 2019b.

Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M., and Lian, D. A Survey on Session-based Recommender Systems. *arXiv:1902.04864 [cs]*, May 2021. URL `http://arxiv.org/abs/1902.04864`. arXiv: 1902.04864.

Wang, X., He, X., Nie, L., and Chua, T.-S. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194, 2017.

Wang, X. and Kadıoğlu, S. Modeling uncertainty to improve personalized recommendations via Bayesian deep learning. *International Journal of Data Science and Analytics*, March 2021. ISSN 2364-415X, 2364-4168. doi: 10.1007/s41060-020-00241-1. URL `http://link.springer.com/10.1007/s41060-020-00241-1`.

Wang, Y., Chan, S. C.-F., and Ngai, G. Applicability of Demographic Recommender System to Tourist Attractions: A Case Study on Trip Advisor. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 97–101, December 2012. doi: 10.1109/WI-IAT.2012.133.

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 495–503, New York, NY, USA, February 2017.

Association for Computing Machinery. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661.3018689. URL `https://doi.org/10.1145/3018661.3018689`.

Wu, C. and Yan, M. Session-aware information embedding for e-commerce product recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management*, pages 2379–2382, 2017.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., and Tan, T. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 346–353, 2019.

Xiao, Y., Yao, L., Pei, Q., Wang, X., Yang, J., and Sheng, Q. Z. Mgnn: Mutualistic graph neural network for joint friend and item recommendation. *IEEE Intelligent Systems*, 35 (5):7–17, 2020.

Xiao, Y. and Ezeife, C. I. E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data. In Ordonez, C. and Bellatreche, L., editors, *Big Data Analytics and Knowledge Discovery*, Lecture Notes in Computer Science, pages 70–82, Cham, 2018. Springer International Publishing. ISBN 978-3-319-98539-8. doi: 10.1007/978-3-319-98539-8_6.

Xue, H.-J., Dai, X., Zhang, J., Huang, S., and Chen, J. Deep matrix factorization models for recommender systems. In *IJCAI*, volume 17, pages 3203–3209. Melbourne, Australia, 2017.

Yap, G.-E., Li, X.-L., and Yu, P. S. Effective Next-Items Recommendation via Personalized Sequential Pattern Mining. In Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., and Yoo, J., editors, *Database Systems for Advanced Applications*, Lecture Notes in Computer Science, pages 48–64, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-29035-0. doi: 10.1007/978-3-642-29035-0_4.

Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., and Wu, J. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

Yu, F., Liu, Q., Wu, S., Wang, L., and Tan, T. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732, 2016.

Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., and He, X. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 582–590, 2019.

Zaki, M. J. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1):31–60, January 2001. ISSN 1573-0565. doi: 10.1023/A:1007652502315. URL `https://doi.org/10.1023/A:1007652502315`.

Zeng, Z., Lin, J., Li, L., Pan, W., and Ming, Z. Next-item recommendation via collaborative filtering with bidirectional item similarity. *ACM Transactions on Information Systems (TOIS)*, 38(1):1–22, 2019.

Zhang, S., Tay, Y., Yao, L., and Sun, A. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018a.

Zhang, S., Yao, L., Sun, A., Wang, S., Long, G., and Dong, M. Neurec: On nonlinear transformation for personalized ranking. *arXiv preprint arXiv:1805.03002*, 2018b.

Zhang, Y. and Cao, J. Personalized recommendation based on behavior sequence similarity measures. In *Behavior and Social Computing*, pages 165–177. Springer, 2013.

Zhao, G., Lee, M. L., Hsu, W., and Chen, W. Increasing temporal diversity with purchase intervals. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 165–174, 2012.

Zheng, N., Song, X., Chen, Z., Hu, L., Cao, D., and Nie, L. Virtually trying on new clothing with arbitrary poses. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 266–274, 2019.

Zhong, M., Li, C., Wen, J., Liu, L., Ma, J., Zhang, G., and Yang, Y. Hignet: Hierarchical and interactive gate networks for item recommendation. *IEEE Intelligent Systems*, 35 (5):50–61, 2020.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Zhu, J., Shan, Y., Mao, J., Yu, D., Rahmanian, H., and Zhang, Y. Deep embedding forest: Forest-based serving with deep embedding features. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1703–1711, 2017.

Zimdars, A., Chickering, D. M., and Meek, C. Using Temporal Data for Making Recommendations. *arXiv:1301.2320 [cs]*, January 2013. URL `http://arxiv.org/abs/1301.2320`. arXiv: 1301.2320.

Vita Auctoris

NAME:	Mahreen Nasir Butt
PLACE OF BIRTH:	Pakistan
YEAR OF BIRTH:	1984
EDUCATION:	Doctor of Philosophy in Computer Science, University of Windsor Windsor, Ontario, Canada, 2022.
	Master of Science in Computer Science, Lahore College for Women University, Lahore, Punjab, Pakistan, 2009
	Bachelor of Science in Computer Science, Lahore College for Women University Lahore, Punjab, Pakistan, 2006