1-1-2023

Semantic enhanced Markov model for sequential E-commerce product recommendation

Mahreen Nasir
University of Windsor

C. I. Ezeife
University of Windsor

Recommended Citation

Nasir, Mahreen and Ezeife, C. I.. (2023). Semantic enhanced Markov model for sequential E-commerce product recommendation. *International Journal of Data Science and Analytics*, 15 (1), 67-91.
https://scholar.uwindsor.ca/computersciencepub/62

International Journal of Data Science and Analytics (2023) 15:67–91
https://doi.org/10.1007/s41060-022-00343-y

REGULAR PAPER

Semantic enhanced Markov model for sequential E-commerce product recommendation

Mahreen Nasir[1] · C. I. Ezeife[1]

Received: 29 June 2021 / Accepted: 24 June 2022 / Published online: 24 July 2022

Abstract
To model sequential relationships between items, Markov Models build a transition probability matrix **P** of size $n \times n$, where n represents number of states (items) and each matrix entry $p_{(i,j)}$ represents transition probabilities from state i to state j. Existing systems such as factorized personalized Markov chains (FPMC) and fossil either combine sequential information with user preference information or add the high-order Markov chains concept. However, they suffer from (i) model complexity: an increase in Markov Model's order (number of states) and separation of sequential pattern and user preference matrices, (ii) sparse transition probability matrix: few product purchases from thousands of available products, (iii) ambiguous prediction: multiple states (items) having same transition probability from current state and (iv) lack of semantic knowledge: transition to next state (item) depends on probabilities of items' purchase frequency. To alleviate sparsity and ambiguous prediction problems, this paper proposes semantic-enabled Markov model recommendation (SEMMRec) system which inputs customers' purchase history and products' metadata (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 (Doc2vec and TF-IDF) which consider the context of items' usage. Next, this extracted knowledge is integrated into the transition probability matrix **P** to generate personalized sequential and semantically rich next item recommendations. Experimental results on various E-commerce data sets exhibit an improved performance by the proposed model

Keywords Recommendation systems · Sequential recommendation · Semantics · Markov model · Collaborative filtering · E-commerce

1 Introduction

Domain-driven data mining discovers actionable knowledge and insights from complex data and behaviors. Various frameworks, algorithms, models and evaluation systems for actionable knowledge discovery have been studied in the past [68]. Traditional data driven pattern mining and knowledge discovery lacks outputs that are actionable. However, in this modern era of big data, it is imperative to discover knowledge and insights from complex data to facilitate business decision makers for performing appropriate actions. For instance, big E-commerce platforms like Amazon[1] and AliBaba[2] strive continuously to discover actionable knowledge (decision-making actions) from their customers' historical trends to better serve their customers' future needs and retain their market share. The past years have seen a significant paradigm shift in the evolution of domain-driven actionable knowledge discovery from the traditional data-driven pattern mining [14,21,22]. During the last decade, several new research problems and challenges emerged where incorporating the domain knowledge into data mining processes and models (e.g., text mining, deep neural networks, graph embedding

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. C. I. Ezeife.

✉ Mahreen Nasir
nasir11d@uwindsor.ca

C. I. Ezeife
cezeife@uwindsor.ca

1 School of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada

1 https://www.amazon.com/.
2 https://www.alibaba.com/.

and reinforcement learning) is important. Research on actionable knowledge discovery can be seen as a shift in paradigm from knowledge discovery from data to actionable knowledge discovery [11,13,15] by mining and extracting complex data patterns for complex knowledge in either a multi-feature, multi-source, or multi-method scenario [12].

Next, we will discuss some preliminaries such as mining sequential patterns from user-item interactions (customers' historical purchases) to gain actionable insights for generating sequential next item recommendation for customers' and then how to enrich this sequential modeling process with E-commerce products' domain knowledge by first extracting items' semantic knowledge based on their (a) usage (e.g., products co-purchased or co-reviewed) and (b) textual features by finding similarity between products according to their characteristics using distributional hypothesis methods (Doc2vec and TF-IDF) which consider the context of items' usage. The semantic and sequential product knowledge is then integrated into the sequential recommendation process to facilitate the domain driven data mining process, hence, providing domain-driven knowledge discovery for semantic and sequential next item recommendation.

1.1 Semantic and sequential E-commerce recommendation

E-Commerce Recommendation Systems (RS) facilitate customers' purchase decision by recommending products or services of interest [2,9,31]. To increase revenue and retain customers' loyalty, designing a recommender system tailored toward individual customers' need is inevitable for retailers. Collaborative filtering (CF), a common recommendation technique, takes user-item interaction matrix as input which represents user interactions (clicks, views, purchases) performed 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 [2,19,51,55,56,60]. CF-based methods suffer from sparsity [7,30,66] and cold start [7,44,66] due to low user-item interaction. Alternately, content based methods, generate recommendations based on the content (features) of the item and suffer from content overspecialization (lack of diversity in recommend-ed products) due to the use of specific features only [1]. Therefore, to generate recommendations that are of interest to user, it is important to create user and item profiles by extracting semantics (meaningful relationships) from user-item interactions (clicks, views, purchases) and item' metadata (title, description, brand). Such input information can better reflect user's interest by capturing more insights about users' behavior. These include user and item information (e.g., metadata) [37,64], information contributed by users (e.g., tags, geotags, multimedia content, free comments, reviews) [40,64,69] and information associated with user-item interaction also known as context [1]. An example of context information can be when a user is interacting with an item, such as purchasing an item on special occasions, watching a movie during a holiday or listening to a song while driving.

Furthermore, as users' preferences tend to change with time, the above methods are unable to adapt their recommendations to meet users' short- and long-term interests. Learning the sequential purchase patterns of user interactions based on the timestamps is 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 provides useful insights about users' behavior.

In the next subsection, we will discuss the process of learning sequential purchase patterns to capture users' short- and long-term interests (Sect. 1.1.1), and then in Sect. 1.1.2, we will present details about enriching this sequential purchase pattern learning process by incorporating the semantic knowledge of products extracted from products' metadata and customers' purchase histories to generate recommendations that are of interest to the user.

Here, we present details of notations and symbols used in the paper in Table 1.

1.1.1 Learning sequential patterns in recommendation systems

Learning sequential associations (relationships) betwe-en 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. Recommendation systems that aim to model and understand sequential user behaviors for next item prediction are sequential recommendation systems. In sequential recommendation systems, sequential patterns of user interactions can be learnt through various approaches such as (i) traditional approaches (sequence similarity, sequential pattern mining), (ii) factorization and latent representation-based methods (matrix factorization and Markov models) and (iii) neural network-based methods (recurrent neural networks) [47].

Mining-based methods such as sequential pattern mining [3,5,7,36,41] mine frequent sequential patterns (where a pattern represent item co-occurrences when the items appeared in the same order) from a sequential database (such as historical purchase or click sequence database of customers) based on a user-defined threshold (minimum support) to learn sequential associations between itemset sequences. A sequence is known as a frequent sequential pattern if an item's support count is greater than the minimum support. The pattern mining process is performed offline, and mined patterns are then represented as a set of association rules. The strength of a rule is determined by values such as support

Table 1 Symbols and notations

Symbol	Description
$\mathbf{P}$	Transition probability matrix
S	Observable states
$p(i, j)$	Entry in transition probability matrix, where each entry represents the probability of going from state i to state j (i.e., probability of purchasing product j after purchasing product i
s_i	Current state
s_j	Next state
$\mathbf{p_s}$	Initial probability distribution vector for a state to be a start state
$\mathbf{p^t}$	Probability vector for transitioning to states at time t
$\mathbf{p}^{t+1}$	Probability vector for transitioning to states at time $t+1$
C	Set of all customers
I	Set of all products
m	Size of set of customers
n	Size of set of all products
H_j	Purchase history of a customer j (ordered set of purchased products
$c_{j,t}$	Products purchased by customer j at time t
t_j	Last timestep when the customer made a purchase
D	Set of all documents
T	Set of all unique tokens in the documents
I	Set of all product descriptions
U	Set of all unique words in product descriptions
$w_{k,j}$	Weight for the term t_k in a document d_j
PF	Product Frequency Matrix
N	Total number of product descriptions
n_k	Number of product descriptions where a term appears at least once
$\mathbf{W}$	Word vector for each word
$\mathbf{D}$	Document vector for each document
$\mathbb{R}$	Set of all nonnegative real numbers
$\mathbf{M}$	Item to Item semantic similarity matrix
V	Vocabulary of all words
U	Weight matrix from input to hidden layer
X	Word-to-Word co-occurrence matrix
Y	Weight matrix from hidden to output layer

and confidence. So, for a given transaction (with items that user has already bought), the goal is to predict next items in sequence that the customer is likely to purchase. These items are predicted by performing a database scan for matching rules and then applying them on the given transaction. For recommendation, the Naïve approach is to find pairwise item co-occurrence frequencies to get purchase recommendations of the form 'Customers who purchased... also purchased...'.

Probabilistic and machine learning-based methods aim to learn sequential patterns by modeling sequential behaviors from past observations to predict future ones, i.e., user actions. Markov Chains [6,10,50,59] are successful probabilistic models that model sequential patterns and are an early approach to Top-K sequential recommendation. Markov chain (MC) model-based RS utilize sequential data (by predicting the users' next action based on the last actions). They are assumed to capture underlying relations from the data by detecting sequential patterns through stochastic transitions (random probability) between states.

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 [52]. For a given set of states $S = \{s_1, s_2, s_3, \ldots, s_n\}$, the process initiates from one of these states, and it moves successively one step toward 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 [52]. 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 [39,52,54].

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 1.1** Markov Model Process for Next Item Recommendation*

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.

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 prod-

$$\mathbf{P}^{t} = \begin{array}{c|c} & \begin{array}{ccc} A & B & C \end{array} \\ \begin{array}{c} \mathrm{A} \\ \mathrm{B} \\ \mathrm{C} \end{array} & \begin{bmatrix} 0.80 & 0.10 & 0.10 \\ 0.20 & 0.70 & 0.10 \\ 0.10 & 0.30 & 0.60 \end{bmatrix} \end{array} \times \mathbf{p_s} = \begin{bmatrix} 0.40 \\ 0.24 \\ 0.36 \end{bmatrix} \rightarrow \mathbf{p}^{t+1} \begin{bmatrix} 0.40 \\ 0.31 \\ 0.28 \end{bmatrix}$$

Fig. 1 First-order Markov model process (transition probability matrix)

ucts) as shown in Fig. 1. 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 (from) and the rows indicate the end state (to). 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 take the 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 $\mathbf{P_{t+1}}$).

The first-order Markov chain is an item-to-item tra-nsition matrix learnt using maximum likelihood estimation (MLE) where the transition to next state depends only on the current state. An L-order Markov chain makes recommendations based on L previous actions. However, including more states increases the computations and adds complexity to the system [52].

Neural network-based approaches such as recurrent neural networks (RNNs) [28,29,48] 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 modeling 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 as well.

1.1.2 Using semantics in sequential recommendation

To improve the effectiveness of intelligent information access platforms (e.g., Recommendations Systems), semantics (meanings) are required to (i) improve the qua-lity of user profiles and (ii) understand the information conveyed by the textual content represented by items' metadata. To understand a target users' needs, semantics can be obtained by utilizing (i) top-down or (ii) bottom-up approaches [18].

Top-down approaches learn semantics by utilizing external knowledge sources (e.g., taxonomies, dictionaries and ontologies) to create user profiles and interpret the meaning of items. The idea is to map the item features with semantic concepts [57], or linking the item to the concepts in the knowledge graph [23,42,58].

Alternately, bottom-up approaches explore paradigmatic (e.g., substitutional words) [26] and syntagmatic (co-occur ring words in the same context) [4] relations between words. An example of paradigmatic relationship is substituting 'Ferrero Rocher' chocolate with 'Ferrero Rocher Heart' due to similarity in brand, ingredients and characteristics. The example, ' I ate fish with chips' represents syntagmatic relationship between 'fish' and 'chips' as most often both are eaten together (same context). Similarly, a user's review stating 'I liked my new laptop' represents syntagmatic relationship between 'laptop' and 'liked.' Bottom-up approaches represent each term in a vector space model (VSM) [63] by creating a Term-Context matrix. The Term-Context matrix encodes the context (e.g., a situation) representing the usage of the term. The context depends upon the granularity which can be coarse grained (e.g., a whole document) or fine grained (a window of words, paragraph or a sentence). For example, a document of customers' reviews regarding the cutlery served and the quality of beverages at the restaurant they dined at can represent a coarse grained context, whereas descriptions of items customers' purchased over the weekend can be an example of a fine grained context. Therefore, a Wordspace (vector space representation of words) is learnt according to the usage of terms in contexts. Terms with similar usage will be similar and represented closely in the vector space. After obtaining these vector representations (e.g., vector representation of item descriptions), similarity calculations such as using cosine similarity between vectors can be performed to find items similar to each other.

To better reflect users' preferences and to interpret the meaning of the items, in this paper, we explored the effectiveness of utilizing semantic knowledge (meaningful relationships between items) learnt through the use of bottom-up models based on distributional hypothesis such as Doc2vec [43] and TF-IDF [53] methods which consider the context of item usage, e.g., their co-occurrence in the purchase sequences to learn the semantic relationships between products (e.g., products co-purchased and co-reviewed along with computing semantic similarities based on their textual features). This semantic knowledge can then be integrated into the Markov process for personalized sequential recommendation process by (i) learning semantic associations between items (ii) creating item transition probability matrix by first extracting the sequential co-occurrences of product pairs, normalizing it and then (iii) fusing the semantic knowledge into the transition probability matrix and using it with users' preferences (personalized vector) to generate semantically similar, sequential next item recommendations. Thus, the inclusion of semantic knowledge can address the issues of sparsity, ambiguous prediction and provide recommendations which are diverse, similar in context (usage) and better reflect users' interests.

1.2 Contributions

The contributions of this study are to improve the performance of recommendation system by fusing products' semantic knowledge (obtained through their usage context and metadata using distributional models) into the transition probability matrix of first-order Markov Models to enrich the process of sequential next item prediction by providing customized next item recommendations to the customer.

In existing studies, top-down approaches were used to extract semantics by utilizing external knowledge sources (e.g., taxonomies, dictionaries and ontologies) to create user profiles and interpret the meaning of items. The idea is to map the item features with semantic concepts [57], or linking the item to the concepts in the knowledge graph [23,42,57] for items (e.g., movies). One such limitations of these approaches is that it requires a pre-defined product ontology which may not be applicable to all retailers as each of them may have a different hierarchical structure (categories) to represent the products in store. Therefore, in this study, we aim to explore the bottom-up approaches to extract products' semantic knowledge extracted from customers' historical records (e.g., purchase histories) and products metadata (e.g., title, brand and description) in terms of their textual features (attributes), concepts (meaning) and their context of usage (e.g., co-purchased, co-reviewed). The extracted semantic knowledge is then utilized to learn (i) semantic and sequential relationships between items, (ii) potential candidate item generation and (iii) generating semantic-rich and sequential next item recommendations for the target customers. The rationale to obtain product embeddings through aggregating two models (e.g., TFIDF [53] and Doc2Vec [43]) on product sequences was to capture semantic information from the purchase sequences at the local and the global level where TFIDF [53] captures information from the purchase sequences at the local level by extracting tokens (key words) present in the products' textual metadata. Doc2vec [43] on the other hand, provides global context as in Doc2Vec model, the words (products) and the paragraph (products' metadata in a purchase sequence) are trained jointly and a document embedding (where a document represents collection of all product descriptions, title, brand in a list of list format and each list element represents description, title and brand of a product purchased, along with a document ID for each document) is generated. The unified joint embedding (hybrid feature product vectors) obtained by averaging vector embeddings acquired from both models (TFIDF [53] and Doc2Vec [43]) for each product in the corpus, therefore, capture information from the purchase sequences at the local and the global level and this unified representation of products' feature vectors provide better results in terms of finding products which are similar in semantics.

Our approach provides a unified structure for generating product recommendations which are semantically rich, sequential and personalized without requiring the use of user item ratings.

In summary, the main contributions of our work are:

1. Proposing a comprehensive model to learn item (product) semantics by utilizing product features or metadata such as product title, description and brands and then incorporating those obtained semantics in the transition probability matrix of Mar-kov model to enrich the next item prediction process by generating Top-K semantic and sequential recommendation without using any explicit information (such as customers' ratings).
2. Enriching the transition probability matrix in the Markov model by integrating semantic knowledge to address the problem of ambiguous prediction, as well as sparsity.
3. Learning products' semantics by training various distributional models and obtained products' representations by: (a) Individually training TF-IDF [53] and Doc2Vec [43] models using more product features such as title, description and brand from customers purchase sequences and products' metadata, where a document represents the collection of products' title, description and brand purchased by the customer sorted according to the time stamp. (b) Utilizing embeddings obtained from TF-IDF [53] and Doc2Vec [43] to create hybrid embeddings for product representation.
4. Providing personalized next item recommendation by computing a personalized vector for the target customer and utilizing it with the transition probability matrix of the first-order Markov model.
5. Proposing a weighted score measure based on semantic similarity and sequential transition probabilities to determine the relationship between products.

6. Performing extensive and detailed experiments using E-commerce datasets such as Amazon dataset under various categories (auto, baby, garden, office, video).

1.3 Problem definition

The aim of this paper is to generate sequential next item recommendations which are personalized and semantically similar to the customers' last recent purchase. *Formally stating:*

Given a set of customers C with size m and the set of all products denoted by I, where the size is $|I| = n$. Customer j has a purchase history H_j, that is an ordered set of products, $\{C_{j,1}, C_{j,t}, ..., C_{j,tj}\}$ where $C_{j,t}$ represents set of products purchased by the customer at time t and t_j represents the last timestamp at which the customer made a purchase. The goal of the system is to predict (recommend) Top-K next item(s) which are personalized, semantically similar and can be purchased sequentially according to the items in target customers' last recent purchase as we assume that the customers' next purchase depends on the products they have purchased so far, more specifically the recent one. This goal is achieved by (i) extracting semantic knowledge of items from items' metadata (title, description and brand) and (ii) customers' purchase histories (co-purchased and co-reviewed products) and then (iii) fusing the semantic knowledge into the transition probability matrix of Markov model to enrich the sequential recommendation process. The rest of this paper is organized as follows. Section 2 summarizes related work about recommendation systems in particular about sequential recommendation using Markov models and semantic-based recommendation using distribution approaches such as vector space models and 2Vec approaches. Section 3 introduces the proposed model in detail along with the architecture. Section 4 and 5 present the experiments and results along with analysis of the model. Finally, Section 6 presents the conclusion and the future work.

2 Related work

Various studies on recommendation systems exist in the literature. We have explored related work in three main categories as (i) general recommendation methods which only include user feedback without considering any sequential user interactions, (ii) sequential recommendation methods which take into account sequential order of user actions and (iii) semantic-based recommendation methods which learn meaningful relationship between items by using distributional models such as vector space models and 2Vec approaches to better understand user's preferences. In this section, we will discuss them, respectively.

2.1 General recommenders

General recommenders (traditional recommenders)-recommend items through modeling the users' general tastes (preferences) based on their historical interactions. Traditional recommendation methods include C-ollaborative filtering [2,19,20,51,55,56,60], matrix factorization [34,35,50], and rule-based approaches [3,36,66]. Recommendations are generated based on using either explicit feedback (ratings, reviews) or implicit feedback (clicks, purchases). The user may not interact with these recommended items in short term (e.g., next purchased item); however, she may consume them eventually during her later purchases. Common recommendation method to achieve this task is matrix factorization (MF)-based collaborative filtering. The idea is to learn user and item latent vectors to discover underlying preferences of user [35]. MF-based methods may utilize explicit feedback and formulate recommendation as rating prediction problem where a user's preference is reflected through their ratings; however, due to low availability of explicit ratings (as users' may not always prefer to rate items after purchase and not necessarily purchase all products out of thousands of available products), recent MF-based methods learn user preferences through implicit feedback by borrowing the idea of the Learning to Rank technique, which is focused on designing an effective objective loss function for optimization [34]. General recommender systems are good at capturing general features of user behavior. However, they cannot adapt their recommendations to incorporate users' recent interactions as they do not take into account sequential purchase patterns of users'. Furthermore, to retain customers and maximize profit for the retailers, it is important to consider diversity in [49,61] recommendations while still maintaining the accuracy of recommendations.

2.2 Sequential recommenders

Sequential recommender systems represent users' interactions (e.g., clicks, views, purchases) as a sequence and predict the next item(s) with which the user is most likely to interact with.

A sequential recommender system views the interactions of a user as a sequence and aims to predict which item the user will interact with next. Sequential recommendation based on probabilistic and machine learning approaches such as Markov models aim to learn sequential patterns by modeling sequential behaviors from past observations to predict future ones, i.e., user actions. Markov Chains are successful probabilistic models that model sequential patterns. Model-based methods apply machine learning techniques during the model training process. They capture underlying relations from the data. This is achieved by computing an item-to-item matrix where items which are nearest (most similar) to the recently interacted item by the user are recommended. An example is Markov chains-based methods [6,10,50,59] which estimate an item-to-item transition probability matrix to predict the probability of the next item given the last interaction of the user [50,59]. The model detects sequential patterns through stochastic transitions between states.

However, 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 [17] 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 datasets are very large.

Other works utilizing Markov chains for sequential recommendation involve the work by [59] 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 in [71]. They investigate how to extract sequential patterns to learn the next state with a standard predictor, e.g., a decision tree. The authors in [50] 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, [27] 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 [32]. 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 [38] 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 [46], was proposed for generating sequential next course recommendation for students by capturing the sequential relationships between different courses.

Recently, great progress have been shown by deep learning, and many new techniques such as recurrent neural networks (RNN) [28,29,48] and graph-based neural networks [25,67,70] have been adapted to sequential recommendation. RNN-based methods embed users' historical interactions into a latent vector to represent their preferences. To improve sequential recommendation by using memory networks, RUM [16] is proposed which explicitly captures sequential patterns at item and feature level. Also, Bayesian-based deep learning methods [65] are proposed to improve personalized recommendations by building deep learning architectures for learning user and item interactions. A graph neural network-based sequential recommender system (SDE-GNN) [25] captures sequential dependencies and item transition relations within sessions for generating accurate next item recommendations. Another recommender system [67] generates unified friend and item recommendation by incorporating a mutualistic mechanism which models mutual relationships between consumption and social behavior of users. A hierarchical and interactive gate network (HIGnet) model for items' rating prediction is proposed [70] 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.

These techniques capture high-order sequential interaction between items and users; however, they usually require a high computational time and are complex.

To address the sparsity and ambiguous prediction problems [52,54] in Markov-based models, in our proposed model, we integrate semantic information into the Markov model, during its creation to provide semantically rich sequential next item prediction. In summary, we are using semantic context to enrich the transition probability matrix in the Markov Model to address the issues of sparsity, ambiguous prediction and provide personalized, semantic-rich sequential next item recommendations.

2.3 Semantic-based recommendation

Research works have introduced to learn item associations (relationships) for recommendation by learning their semantics (meanings) through various distributional models such as Word2Vec [43], Prod2Vec [24], Glove [45], Doc2Vec [43] and count-based model such as TF-IDF [53]. Semantics 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 such as recommendation systems. Word2Vec [43] creates word embeddings (dense numeric representations of words which stores the contextual (semantic) information in a low-dimensional vector). Using word embeddings, words similar in context tend be to closer to each other in the vector space. The model is basically a two layer neural network architecture based on distributional hypothesis for learning words' semantic representation. Word2vec model is fed with an input of a text corpus, and it outputs a set of feature vectors (embeddings) of dimension d that represent the words in the input corpus. However, these features representing those words are not explicitly defined. Given a target word, the skipgram approach in Word2Vec [43] predicts the probability of a word being a context word. The context of a word w within a sentence is the set of x surrounding words. For example, for the sentence 'My new laptop fell out of my bagpack and broke its screen,' we have the target word 'laptop.' With a sliding window of size=1, and the target word = 'laptop' moving along a sentence, the skipgram model will predict the probability of a word being a context word where a context word is on each side of the target word within the sliding window.

GloVe [45] obtain words' vector representation bas-ed on their co-occurrence information (for example, in a large text corpora, how frequently they appear together). It is observed that certain words co-occur more often with certain words than others. For example, the word 'butter' may occur more frequently with 'cheese' as they share similar context (breakfast items). Given a document with some word w_i, a context window of size n surrounding the word w_i, then there are the subsequent words in that document, i.e., words $w_{i+1}...w_{i+n}$, the model builds a co-occurrence matrix X (a symmetric word-by-word matrix) in which x_{ij} represents number of times word w_j appears inside w_i window among all documents and then from that matrix, the model learns the vector representations of words. The main intuition behind the model is that ratios of word–word co-occurrence probabilities encode some form of meaning.

TF-IDF [53] approach use keyword matching or vector space model (VSM) to represent items (for example, products, music, videos, documents). VSM represents each item in an n-dimensional vector space referred to as spatial representation of items where each dimension in the vector represents a term from the vocabulary obtained from a given collection of items. For example, given a large collection of text documents, a vocabulary will represent all unique words or terms (concepts) that are relevant to the content of the document. Given $D = \{d_1, d_2, \ldots, d_N\}$ denoting a set of documents and $T = \{t_1, t_2, \ldots, t_N\}$ be the vocabulary that is the set of words in the documents where T is obtained by applying some standard natural language processing operations (for example, tokenization, stop words removal and stemming), each document d_j is then represented as a vector in a 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 term vectors, next task is to measure the similarities between feature vectors of documents in order to find documents similar to the target. To achieve this goal, TF-IDF weighting (Term Frequen-cy-Inverse Document Frequency) [53] is used which co-mputes a term's weight as a product of term frequency (TF) weight and inverse document frequency (IDF) weight, where TF represents the number of times the term occurs in the document and IDF represents how rare a term is in the collection. The TF-IDF weight increases if the term occurs frequently within the document and with the term being rare in the collection.

SEMSRec [44] proposed to integrate e-commerce products' semantic and sequential relationships extrac-ted from customers' purchase histories into CF's item similarity matrix to provide semantically similar and sequential recommendations. However, the limitation is that they did not include any item metadata for learning products' semantic knowledge.

3 Architecture of the proposed semantic-enabled Markov model for sequential E-commerce product recommendation (SEMMRec)

We propose a component-based architecture for our proposed system. Figure 2 presents the diagram depicting the architecture of the system. The system consists of four main modules which are (i) data preprocessing (Sect. 3.1), (ii) items' semantic representation (Sect. 3.2), (iii) Markov model building (Sect. 3.3) and (iv) semantic and sequential recommendation module (Sect. 3.4). The dashed lines in Fig. 2 show each module along with the sub-processes being performed in that module. Next, we will discuss each module in detail.

3.1 Data preprocessing module

This module preprocesses the products' metadata (items' title, description and brands) and users' (customers') historical purchase data to be fed as input to the system. The customers' historical data represents customers' purchases made over a period of time. This historical data are processed to be cleaned and then transformed to make it suitable for next modules. The various preprocessing operations involve (i) filling mis-sing values, (ii) removing duplicate records, (iii) sorting and grouping customers' purchase sequences according to the timestamp. Additional preprocessing for products' metadata involves natural language processing operations (NLP) such as (i) tokenization (the process of segmenting text into words, clauses or sentences such as separating words and removing punctuations), (ii) stop words removal (removal of commonly used words unlikely to be useful for learning such as a, the, of), and (iii) stemming which involves reducing related words to a common stem such as reducing the words 'liked,' 'likeliness' and 'likely' to the word 'like' obtained from customer's review about a product expressing her preferences toward the product. All these operations are

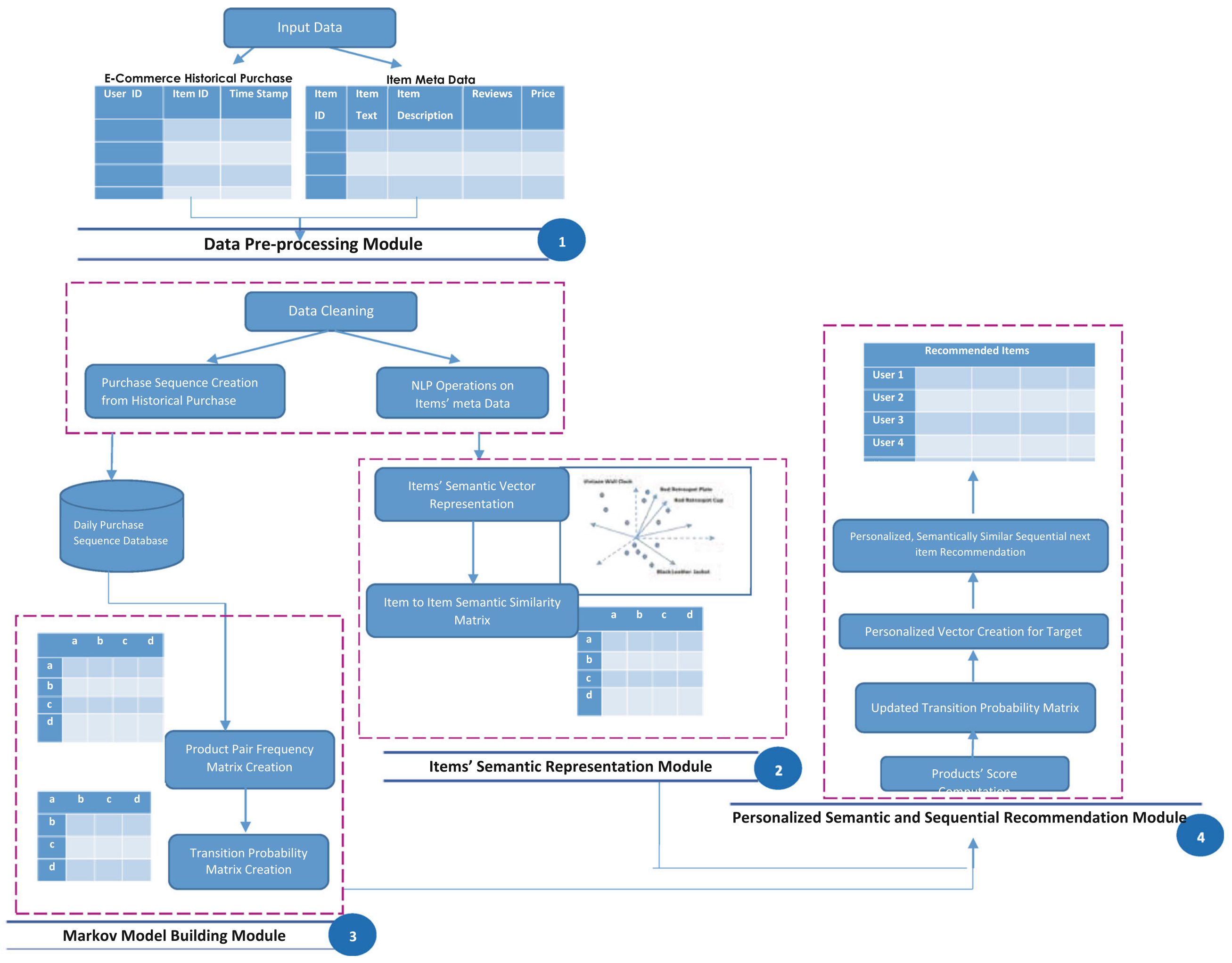

Input Data
E-Commerce Historical Purchase
User ID
Item ID
Time Stamp
Item Meta Data
Item ID
Item Text
Item Description
Reviews
Price
Data Pre-processing Module
1
Data Cleaning
Purchase Sequence Creation from Historical Purchase
NLP Operations on Items' meta Data
Daily Purchase Sequence Database
Items' Semantic Vector Representation
Item to Item Semantic Similarity Matrix
Items' Semantic Representation Module
2
Product Pair Frequency Matrix Creation
Transition Probability Matrix Creation
Markov Model Building Module
3
Recommended Items
User 1
User 2
User 3
User 4
Personalized, Semantically Similar Sequential next item Recommendation
Personalized Vector Creation for Target
Updated Transition Probability Matrix
Products' Score Computation
Personalized Semantic and Sequential Recommendation Module
4

Fig. 2 Architecture of the proposed model SEMMRec

performed before the data can be input to the modules for learning products' semantic representations.

3.2 Items' semantic vector representation (embeddings) module

This module involves learning the product vector representations (semantics) from the customers' purchase histories which are then used in the later phases for computing product similarities, computing each products' score (based on its semantic and sequential occurrence) and then fusing this semantic and sequential information with the transition probability matrix to generate next item recommendation. This module consists of two steps which are (i) learning product semantic vector representation through TF-IDF [53], Doc2Vec models [43], their hybrid and (ii) creating item to item semantic similarity matrix based on learnt semantic vector representation in (i). Next, we discuss the steps in detail.

3.2.1 Items' semantic vector representations

In this subsection, we will present the details of obtaining product's vector representations based on products' metadata using the Doc2vec [43] and TFIDF [53] methods. To explore the impact of obtaining product semantics from other product features (textual data), we create (i) 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 (ii) documents for Doc2vec [43] (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). Additionally, we obtained unified hybrid feature product vectors acquired after training both models Doc2Vec [43] and TF-IDF [53]. This was achieved by averaging vector embeddings acquired from both models for each product in the corpus.

Table 2 Sample historical product purchase records

Customer ID	Stock code	Invoice no.	Invoice date
17850	20674	536365	12/1/10 8:26
17850	21242	536365	12/1/10 8:26
17850	20675	536365	12/1/10 8:26
17850	21245	536365	12/1/10 8:28
17850	20677	536365	12/1/10 8:28
17850	20655	536365	12/1/10 8:30
17850	20677	536365	12/1/10 8:30

To explain the input format and the working of the models, TF-IDF [53] and Doc2Vec [43], we will use sample data from historical product purchase records of customers (Table 2) and products' metadata (Table 3). For both models, we will obtain a semantic vector representation of items across d dimensions (d=100), where an item can represent documents (for example, products, articles, books, customer reviews or product descriptions). For the sake of simplicity, we represent products using small alphabets instead of their id's as $\{a, b, c, d, e, f\}$ where 'a' = product with stock code 20674, 'b'= 21242, 'c' = 21238,'d'=21245, 'e'=21239, 'f' =20655 and 'g'='20675.' Next, consider Table 4 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 purchase 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. Next, we will present an

Table 4 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)

example to explain the process of learning products' semantic vector representation using TF-IDF [53] and Doc2Vec [43] models.

Example 3.1 Products Semantic Vector Re-presentations Using TF-IDF

Problem:

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

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

I= ['airline bag vintage jet set brown,' 'wood land charlotte bag,' 'black candelabra tlight holder,' 'medium ceramic top storage jar'], $T = \{t_1, t_2, ..., t_N\}$, where in our case, U= ['`lue,' 'bowl,' 'dot,' 'green,' 'luggage,' 'pink,' 'plate,' 'polka,' 'queen,' 'red,' 'retrospot,' 'skies,' 'tag']

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

Table 3 Sample of product metadata

Stock code	Title	Description	Brand
20655	Queen of skies Luggage Tag	Suitcase tag made PU material, in front with a protective film, waterproof. Fully Bendable and Flexible Material to Prevent Breaking or Losing Your leather luggage tags	PAGSRAH
20674	Green polka Dot bowl	Earthenware, largest measures 5.5 in h × 12 in l × 11.25 in hand wash	Tag limited
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 in h × 12 in l × 11.25 in. 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 and Creative innovations	

Table 5 Frequency count of unique tokens occurring in the product descriptions

Products	Tokens												
	blue	bowl	Dot	green	luggage	pink	plate	polka	queen	red	retrospot	skies	tag
Queen of skies luggage tag	0	0	0	0	1	0	0	0	1	0	0	1	1
Green polka dot plate	0	0	1	1	0	0	1	1	0	0	0	0	0
Pink polka dot bowl	0	1	1	0	0	1	0	1	0	0	0	0	0
Red retrospot plate	0	0	0	0	0	0	1	0	0	1	1	0	0
Blue polka dot bowl	1	1	1	0	0	0	0	1	0	0	0	0	0
Green polka dot bowl	0	1	1	1	0	0	0	1	0	0	0	0	0

Table 6 Term frequencies (TF Computation)

Products	Tokens												
	blue	bowl	dot	green	luggage	pink	plate	polka	queen	red	retrospot	skies	tag
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
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
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
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
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
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

Table 5 shows frequency count of thirteen terms occurring in the product descriptions For example, the term 'plate' appears two times among the product descriptions (as indicated by a 1 in each corresponding row where the term appears in a product). The steps to learn the product feature vectors using TF-IDF are given below:

Step 1: Term Frequency Computation (TF)

Term Frequency (TF) will be computed by using Eq. (1),

$$\text{TFIDF}\left(t_k, d_j\right) = TF\left(t_k, d_j\right) \cdot \log \frac{N}{n_k} \tag{1}$$

where TF(t_k, d_j) represents frequency of term t_k in document d_j and is represented as f_{t_k,t_j} and the maximum is computed the frequencies $f_{z,j}$ of all terms t_z that occur in document d_j using Eq. (2),

$$TF\left(t_{k,}d_j\right) = \frac{f_{t_k,d_j}}{\max_z f_{z,j}} \tag{2}$$

For example term frequency for 'pink' in product description 'blue polka dot bowl' is computed as:

TF (pink, blue polka dot bowl) = 0/4 =0.00. Similarly, the term frequencies of all tokens are shown in Table 6.

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 7 shows the IDF of all terms.

Table 7 IDF computation

Term	IDF	Term	IDF	Term	IDF	Term	IDF
blue	0.78	green	0.48	plate	0.48	red	0.78
bowl	0.30	luggage	0.78	polka	0.18	retrospot	0.78
dot	0.18	pink	0.78	queen	0.78	skies	0.78
						tag	0.78

Step 3: TF- IDF Computation

The TF-IDF is computed using Eq. (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 'pink' and 'bowl' in the description 'blue polka dot bowl' will be:

TF-IDF (pink, blue polka dot bowl) = TF (pink) × IDF (pink, blue polka dot bowl) = 0.00 × 0.78 = 0.00 TF-IDF (bowl, blue polka dot bowl) = TF (bowl) × IDF (bowl, blue polka dot bowl) = 0.25 × 0.30 = 0.08 Table 8 shows the TF-IDF's of all tokens.

After the computation of TF-IDF, we can represent each product as a d-dimensional feature vector, where each dimension represents a feature (token). For our example, we have thirteen features (tokens) and the third row in Table 8 represents the product 'pink polka dot bowl' as term weighted vector. 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

Table 8 TF-IDF of tokens in product descriptions (product vectors using TF-IDF)

Products	Tokens											
	blue	bowl	dot	green	luggage	pink	plate	queen	red	retrospot	skies	tag
Queen of skies luggage tag	0.00	0.00	0.00	0.00	0.19	0.00	0.00	0.00	0.19	0.00	0.00	0.19
Green polka dot plate	0.00	0.00	0.04	0.12	0.00	0.00	0.12	0.04	0.00	0.00	0.00	0.00
Pink polka dot bowl	0.00	0.08	0.04	0.00	0.00	0.19	0.00	0.04	0.00	0.00	0.00	0.00
Red retrospot plate	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.00	0.26	0.26	0.00
Blue polka dot bowl	0.19	0.08	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00
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

$$PV_{tfidf} = \left[\begin{array}{lllllll} & F0 & F1 & \multicolumn{3}{c}{\dots\dots\dots} & F98 & F99 \\ a & 0.075502 & -0.026424 & \vdots & \vdots & \vdots & -0.073289 & -0.055986 \\ b & 0.202582 & -0.226119 & \vdots & \vdots & \vdots & -0.028236 & 0.037867 \\ c & 0.080445 & -0.041201 & \vdots & \vdots & \vdots & -0.030099 & -0.067652 \\ d & 0.074386 & -0.036098 & \vdots & \vdots & \vdots & -0.076671 & 0.041561 \\ e & 0.114951 & -0.034461 & \vdots & \vdots & \vdots & -0.047242 & -0.070353 \\ f & 0.000589 & -0.001437 & \vdots & \vdots & \vdots & 0.004696 & 0.003122 \end{array}\right]$$

Fig. 3 Product vectors obtained from TF-IDF model

$$PV_{\mathrm{doc}\ 2vec} = \left[\begin{array}{llllllll} & F0 & F1 & \multicolumn{3}{c}{\dots\dots\dots} & F98 & F99 \\ a & -0.729392 & -0.694623 & \vdots & \vdots & \vdots & 0.784198 & -0.709698 \\ b & 0.478108 & 0.810839 & \vdots & \vdots & \vdots & 0.128663 & -0.818151 \\ c & 1.560531 & 0.965368 & \vdots & \vdots & \vdots & -0.794381 & -0.585287 \\ d & -0.309573 & 0.418730 & \vdots & \vdots & \vdots & 0.743325 & 0.427864 \\ e & -0.100401 & 1.024751 & \vdots & \vdots & \vdots & -0.044198 & 0.284042 \\ f & 0.685829 & 0.646765 & \vdots & \vdots & \vdots & 1.036193 & 1.161657 \end{array}\right]$$

Fig. 4 Product vectors obtained from Doc2vec model

$$PV_{d2vec\&tfidf} = \left[\begin{array}{llllllll} & F0 & F1 & \multicolumn{3}{c}{\dots\dots\dots} & F98 & F99 \\ a & -0.034295 & 0.053539 & \vdots & \vdots & \vdots & -0.121241 & 0.018026 \\ b & 0.042981 & -0.004462 & \vdots & \vdots & \vdots & -0.061913 & 0.104531 \\ c & -0.022603 & 0.041644 & \vdots & \vdots & \vdots & -0.068960 & -0.068960 \\ d & -0.021472 & 0.072360 & \vdots & \vdots & \vdots & -0.128279 & 0.069797 \\ e & -0.007805 & 0.065704 & \vdots & \vdots & \vdots & -0.082256 & 0.027480 \\ f & -0.038842 & 0.030226 & \vdots & \vdots & \vdots & 0.031924 & 0.204722 \end{array}\right]$$

Fig. 5 Hybrid product vectors by Doc2vec TF-IDF model

product vector matrix across 100 dimensions (features) as shown in Fig. 3 where each column represents the feature and each rows represents the product across 100 dimensions.

Example 3.2 Learning Product Vector Representations Using Doc2vec *Problem:*

Given a collection of documents (collection of product descriptions, product title and product brand in a list of list format 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 plate add a splash of color bright party detail green white dots dessert plates party2u], [red retrospot plate beautiful plates composed high rat-ed heavy weight plastic material render plate leak free soak resistant cut proof unbreakable, silver spoon], [gr-een polka dot bowl earthenware largest measures hand wash tag limited]]

Intermediates: vectors (word 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: Each document with a vector representation of across N dimensions.

Steps: Doc2Vec process to learn document representation is similar to 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. 4.

We then created a hybrid matrix of product vector representations as hybrid of PV_{tfidf} and $PV_{doctovec}$ (Fig. 5) to better learn product semantics. This is achieved by averaging the embeddings of each product in the respective matrices.

3.2.2 Item-to item semantic similarity matrix creation

Next step is to create the item to item semantic similarity matrix **M** to compute products' semantic similarity by applying cosine similarity on product vectors in the $PV_{doc2vectfidf}$

$$M = \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 1.00 & 0.84 & 0.88 & 0.80 & 0.81 & 0.93 & 0.98 \\ b & 0.84 & 1.00 & 0.91 & 0.86 & 0.78 & 0.02 & 0.87 \\ c & 0.88 & 0.91 & 1.00 & 0.96 & 0.98 & 0.95 & 0.94 \\ d & 0.80 & 0.86 & 0.96 & 1.00 & 0.78 & 0.35 & 0.86 \\ e & 0.81 & 0.78 & 0.98 & 0.77 & 1.00 & 0.10 & 0.89 \\ f & 0.93 & 0.02 & 0.95 & 0.35 & 0.10 & 1.00 & 0.45 \\ g & 0.98 & 0.87 & 0.94 & 0.86 & 0.89 & 0.45 & 1.00 \end{array}\right]$$

Fig. 6 Item-to-Item Semantic Similarity Matrix M

$$PF = \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.25 & 0.50 & 0.5 & 0.75 & 0.25 & 0.25 \\ b & 0.50 & 0 & 0.25 & 0.25 & 0.75 & 0 & 0.5 \\ c & 0 & 1.00 & 0 & 0 & 0.25 & 0 & 0 \\ d & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ e & 0 & 0.25 & 0.5 & 0 & 0 & 0 & 0 \\ f & 0 & 0. & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Fig. 7 Product frequency matrix (PF)

matrix using Eq. (3),

$$\text{Cosine Similarity } (x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3}$$

where x_i and y_i represent components of vectors for products x and y, respectively. For example, to compute the similarity of product 'a' (Green Polka Dot Bowl) with product 'e' (Pink Polka Dot Cup) and product 'g' (Blue Polka Dot Bowl), we will take their corresponding product vectors (column) from the hybrid PV matrix and compute cosine similarity between them.

So, Cosine similarity (a,e) is 0.81 and Cosine similarity (a,g) is 0.98, which shows that product 'a' is more close to product 'g' in the vector space than product 'e.' Similarity between other products is computed in the same way. Next, populate the item–item similarity matrix **M** using Eq. (4). Each entry $m_{i,j}$ in the matrix **M** represents semantic similarity between products i and j in the vector space. Figure 6 shows a sample of matrix **M**.

$$\mathbf{M}_{i,j} = \begin{cases} 1, & \text{if } i = j \\ \text{CosineSimilarity } (i, j), & \text{otherwise} \end{cases} \tag{4}$$

3.3 Markov model building module

This module builds 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. Next, 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 as first-order, each state is formed by a single action, i.e., a customer purchased a product. In the case of $L-th$-order models (where L represents the order), the state-space will correspond to all possible sequences of L 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 **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.

3.3.1 Product pairs frequency matrix creation

Based on the historical purchase information of the customers (purchase sequences from Table 4, we first compute product frequency, PF matrix, which is an $n \times n$ matrix where each entry $pf_{i,j}$ 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 in Table 4 is shown in Fig. 7. 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, $f_{a,b} = \frac{1}{4} = 0.25$. The '0' in the matrix shows that these pairs of products are never purchased together.

3.3.2 Transition probability matrix creation

After we compute the frequencies of matrix PF, we need to normalize it to get **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. (5), 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. 8 shows the transition probability matrix **P** created after normalizing

$$\mathbf{P} = \left[\begin{array}{cccccccc} & \mathrm{a} & \mathrm{b} & \mathrm{c} & \mathrm{d} & \mathrm{e} & \mathrm{f} & g \\ \mathrm{a} & 0 & 0.11 & 0.22 & 0.22 & 0.33 & 0.11 & 0.10 \\ \mathrm{b} & 0.22 & 0 & 0.11 & 0.11 & 0.33 & 0 & 0.22 \\ \mathrm{c} & 0 & 0.8 & 0 & 0 & 0.20 & 0 & 0 \\ \mathrm{d} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \mathrm{e} & 0 & 0.33 & 0.66 & 0 & 0 & 0 & 0 \\ \mathrm{f} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Fig. 8 Transition probability matrix P

the product frequency matrix PF in Fig. 7.

$$\mathbf{P}_i = \frac{PF_i}{\sum_{j=1}^{n} PF_{i,j}}, \text{ if } \sum_{j=1}^{n} PF_{i,j} > 0 \tag{5}$$

3.4 Personalized semantically rich and sequential next item recommendation module

This module is responsible to generate personalized, semantically rich sequential next item recommendation for the target customer and consists of four steps which are (i) score computation for products, (ii) updating the transition probability matrix in Markov model, (iii) personalized vector creation for target customer and (iv) generating recommendations. Next, we will provide details of each step.

3.5 Score computation for products

Next, enrich the transition probability matrix **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) and then (ii) updating the transition probability matrix entries by using Eq. (7).

$$\text{Score }(i, j) = \alpha(\text{ CosineSimilarity }(i, j)) + \beta(\mathbf{P}(i, j)) \tag{6}$$

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

$$\text{Score }(a, b) = \alpha(\text{ CosineSimilarity }(a, b)) + \beta(\mathbf{P}(a, b))$$
$$\text{Score }(a, b) = 0.80 * 0.84 + 0.2 * 0.11 = 0.69$$

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

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

Fig. 9 Semantic and sequentially rich updated transition probability Matrix P

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

Fig. 10 Normalized semantic and sequentially rich transition probability matrix P

3.6 Semantic and sequentially rich transition probability matrix

The entries in the matrix **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). Figure 9 shows semantic and sequentially rich updated transition probability matrix **P** populated using Eq. (7) after score calculations for sample products. Matrix **P** can now be used to recommend next personalized semantic and sequential items to users (Sect. 3.6.1). 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).

$$\mathbf{P} = \begin{cases} 0, & \text{if } i = j \\ \text{Score }(i, j), & \text{otherwise} \end{cases} \tag{7}$$

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. 8 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. 9 where both products b and f have different probabilities stating that product f has higher probability of being purchased than product b. After normalization, we have the final semantic and sequential transition probability matrix as shown in Fig. 10.

3.6.1 Personalized semantically rich and sequential next item recommendation

For next item recommendation, a Markov Model can be traversed as a random walk [8]. 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 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 $\mathbf{p} \in \mathbb{R}^{1\times n}$ at time t, where $\mathbf{p}$ is a row vector with nonnegative components whose sum equals 1, and $\mathbf{p}_x$ being the probability of starting at vertex x, $\mathbb{R}$ representing set of nonnegative 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 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. (8). 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.

$$\mathbf{p^t} = \begin{cases} 1/\left|I_{j,tj}\right| & \text{if } i \in I_{j,tj} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

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 $\mathbf{P}$, we can walk over the Markov model to obtain $\mathbf{p^{t+1}}$ (the row vector of probabilities at time $t+1$) using Eq. (9) 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.

$$\mathbf{p^t P} = \mathbf{p^{t+1}} \tag{9}$$

***Example 3.3** Semantic and Sequential Next Item Recommendation with Single item purcha-se at time t*

For example, if a customer's 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. (9), we take product of the enriched transition probability matrix and the initial probability vector to get $\mathbf{p}^{t+1}$ as shown in Fig. 11.

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. (10),

$$\mathbf{pP^t} = \mathbf{p^{t+1}} \tag{10}$$

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,

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

Fig. 11 Computation for next item purchase prediction with single item purchase at time t

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \left(\begin{bmatrix} \begin{array}{c|ccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array} \end{bmatrix} \right)^3 . = \begin{bmatrix} 0.151 & 0.145 & 0.160 & 0.141 & 0.150 & 0.091 & 0.158 \end{bmatrix}$$

Fig. 12 Computation to predict next product purchase at time $t + 1$ with single item purchase at time t

Table 9 Product recommendation by proposed semantic enhanced transition probability matrix in Markov model method with single item in users' last purchase record

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 retrospot plate)	a (Green polka dot bowl)	b (Red retrospot plate)	c (Red retrospot cup)
e (Pink polka dot bowl)	c (Red retrospot cup)	e (Pink polka dot bowl)	c (Red retrospot cup)

$$\mathbf{pP^3} = \mathbf{p}^4$$

as shown in Fig. 12.

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 9. 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 3.4 Semantic and Sequential Next Item Recommendation with multiple items purchase at time t

For example, if a customer's 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 $[\frac{1}{7}\,\frac{1}{7}\,\frac{1}{7}\,0\,0\,0\,0]$, we take product of the enriched transition probability matrix and the initial probability vector using Eq. (9) to get $\mathbf{p^{t+1}}$ as shown in Fig. 13.

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. (10) to get, $\mathbf{pP}^3 = \mathbf{p}^4$ as shown in Fig. 14.

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 10. Results are shown for some other products as well.

4 Experimental evaluation

In this section, we present our experimental setup and then results and analysis.

4.1 Datasets and implementation details

Amazon:[3] This dataset includes reviews (ratings, text, helpfulness votes, timestamps), product metadata (descriptions, category information, price, brand, and image features), links (also viewed/also bought graphs). The statistics for dataset are shown in Table 11.

To test our model, we selected the review-K core (which is a subset of the dataset 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 [27,50], the explicit feedback is converted to implicit feedback by setting a rating score of '1' for a user item 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 [27,32,33,50] was used to maintain their models' accuracy.

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

$$\left[\tfrac{1}{7}\ \tfrac{1}{7}\ \tfrac{1}{7}\ 0\ 0\ 0\ 0\right] * \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array}\right] = \left[0.0494\ 0.0447\ 0.0518\ 0.0720\ 0.0744\ 0.0484\ 0.0784\right]$$

Fig. 13 Computation for Next Item Purchase Prediction with multiple items purchase at time t

$$\left[\tfrac{1}{7}\ \tfrac{1}{7}\ \tfrac{1}{7}\ 0\ 0\ 0\ 0\right] * \left[\begin{array}{cccccccc} & a & b & c & d & e & f & g \\ a & 0 & 0.160 & 0.170 & 0.150 & 0.160 & 0.170 & 0.190 \\ b & 0.198 & 0 & 0.20 & 0.196 & 0.190 & 0.006 & 0.204 \\ c & 0.155 & 0.159 & 0 & 0.168 & 0.181 & 0.170 & 0.166 \\ d & 0.166 & 0.176 & 0.197 & 0 & 0.161 & 0.073 & 0.228 \\ e & 0.190 & 0.187 & 0.247 & 0.165 & 0 & 0.022 & 0.192 \\ f & 0.251 & 0.003 & 0.258 & 0.095 & 0.271 & 0 & 0.122 \\ g & 0.189 & 0.215 & 0.182 & 0.165 & 0.162 & 0.087 & 0 \end{array}\right]^3 = \left[0.0646\ 0.0587\ 0.0590\ 0.0680\ 0.0640\ 0.0361\ 0.0666\right]$$

Fig. 14 Computation for next item purchase prediction with multiple items purchase at time $t + 1$ (after 3 time steps)

Table 10 Product recommendation by our proposed method with multiple items' in users' last purchase record

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)

4.2 Preprocessing and hyper-parameter tuning

For dataset 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). In the temporal user splitting, we used train and test splits of (a) 70%, 30% and (b) 80%, 20%. 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 were preprocessed to create train and test data. For leave-one-out, the training data were 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.

Users with purchasing records greater than five are selected. Furthermore, to evaluate the impact of proposed model (SEMMRec's) performance on handl-ing sparse data, two variants of each dataset (auto, baby and garden) were created. For example, the dataset Auto1 refers to dataset 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. 17. All the models have some parameters to tune. We followed 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 = 20. For our model, products' embedding

Table 11 Dataset statistics

Dataset	Amazon				
	No. of users	No. of items	Total inter-actions	Avg. inter-action per user	Avg. inter-action 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

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 $\beta = 0.2$.

4.3 Evaluation metrics

The model was evaluated on commonly used metrics including Recall@K and NDCG@K as used in [33,62,65]. 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:** It is defined as the proportion of relevant items found in the Top-K recommendations.

$$\text{Recall@K} = \frac{\left|R \cap \hat{R}_{1:K}\right|}{|R|} \tag{11}$$

– **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:

$$\text{NDCG@ K} = \frac{\sum_{k=1}^{K} \frac{I_{c_j}^{k}}{\log_2(k+1)}}{\sum_{k=1}^{m_{c_j}^{K}} \frac{1}{\log_2(k+1)}} \tag{12}$$

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.

4.4 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.

4.5 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 12) shows the performance comparison for each model:

1. **Bayesian 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 Model (FISM).** Based on one of the latest recommendation algorithms to capture the relationship between items for personalized recommendations.
3. **Factorized Personalized Markov Chain.** A hybrid approach (FPMC) 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 analyzing 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.
6. **GRU4Rec.** To model sequential dependencies and making predictions in session-based recommendation systems, [28] proposed this method based on recurrent neural networks (RNN's).
7. **Convolutional Sequence Embedding (Caser).** A convolutional neural network (CNN)-based meth-od which takes the embedding matrix of the L most recent items and applies convolution operations on it to achieve sequential recommendation [62].
8. **Bert4Rec.** A sequential recommendation model with bidirectional encoder representations from transformer by [61]. The model utilizes deep bidirectional self-attention

Table 12 Performance comparison of proposed model (SEMMRec) with the baselines

Models\Metric	Datasets											
	Auto		Baby		Garden		Office		Video		Average	
	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

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.

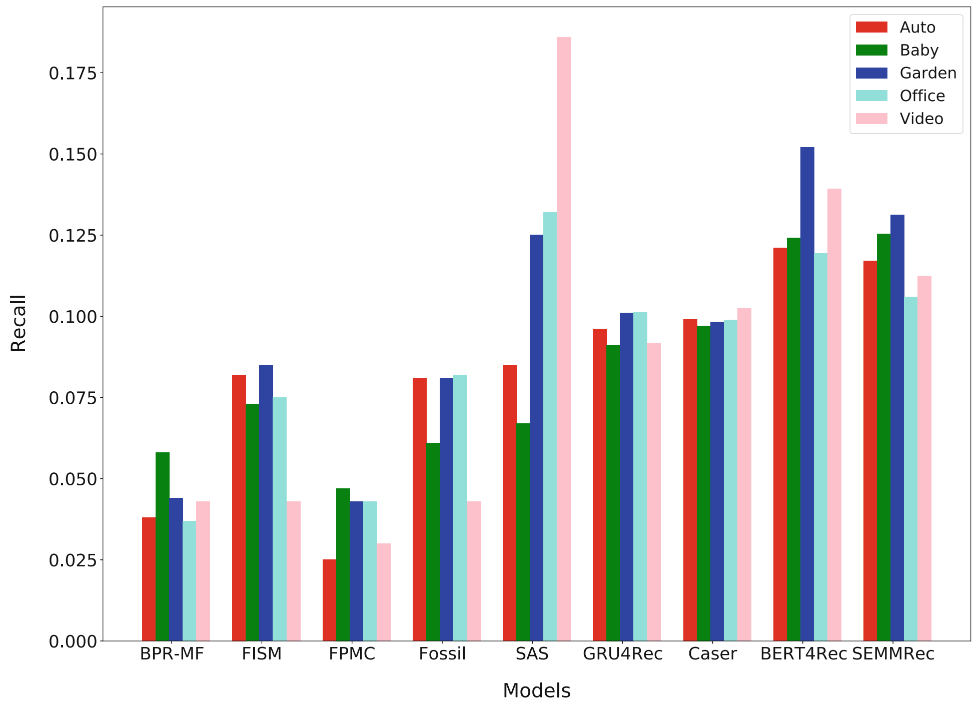
4.6 Results and analysis

Our proposed model SEMMRec gave improved performance in comparison to the baselines on all K tested after incorporating products' metadata 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 of proposed model (SEMMRec) on five datasets in Amazon data (Auto, Baby, Office, Garden, Video) are presented in Table 12. Here, we report results on all evaluation metrics at a cutoff of K=30 where K $\in$ {1, 5, 10, 20, 30, 50, 100}. We notice that SEMMRec performed considerably well on all datasets. A graphical representation of results in Table 12 is also shown in Fig. 15, 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.

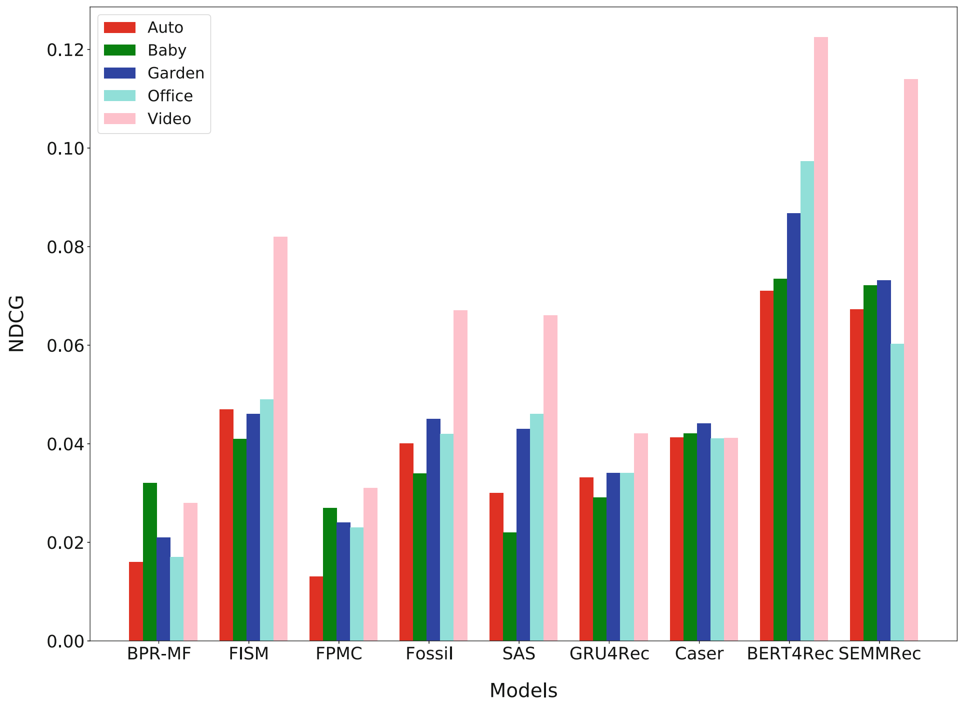
Fossil and FPMC present the user preference and the sequential patterns for the recommender system. However, fossil highlights the sequential pattern better than FPMC, and introduces the concept of FISM. In this regard, as presented in Table 12, Fossil outperforms FPMC on all the datasets; the average of Recall is 0.069 and NDCG 0.045 indicating that learning sequential information about customers' behavior 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 deals 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 methods outperforms almost all conventional recommendation algorithms. However, they require more hyper-parameter tuning and training the model takes much longer. Additional recommender systems such as GRU4Rec [28], Caser [62] and Bert4Rec [61] are also presented for baseline comparisons, and the results show that except for BERT4Rec, our model (SEMMRec) has shown good performance in comparison with other existing related approaches. SEMMRec's performance in comparison with BERT4Rec is bit low, and we believe that this is because BERT4Rec is a bidirectional transformer-based model which learns a bidirectional representation model and makes recommendations by allowing each item in users' historical behavior to integrate information from both left and right sides, whereas our proposed SEMMRec model is probabilistic based (Markov Model), a unidirectional model that only learns the representations from user's historical behavior and items' metadata. Figure 15 illustrates the NDCG and

Fig. 15 Performance comparison of proposed SEMMRec with baseline models on five datasets on the basis of **a** Recall and **b** NDCG



(a) Recall



(b) NDCG

Table 13 Effect of different order of L in Markov chain in SEMMRec

	Metric	Order of L				
Dataset		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

recall evaluations of the five datasets for all the methods used in the experiment. According to these results, SEMM-Rec has an improved average recall and NDCG compared to all presented baseline recommendation methods; however, it does has a slight low Recall and NDCG in comparison with BERT4Rec as mentioned above.

Given performance of all existing methods, it can be said that SEMMRec mostly has better recommendation performance than other related recommendation models as it extract semantic knowledge of items from items' metadata (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.

Next, we will discuss the impact of different order of Markov chain (L), effect of sparsity and effects of train and test split on our proposed SEMMRec Model.

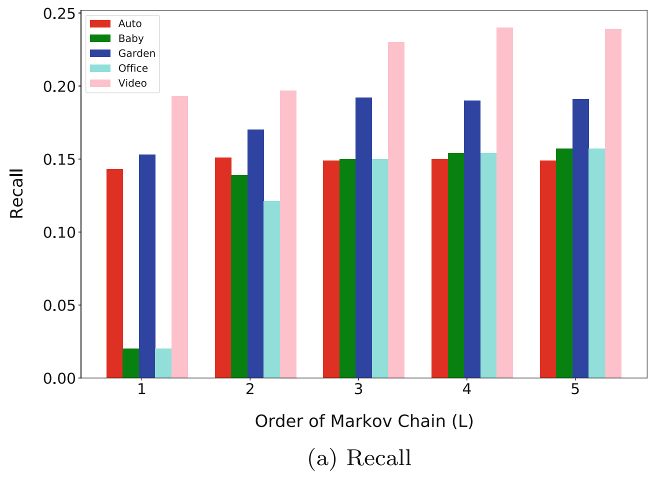
4.6.1 Influence of different order L of Markov chain

Next, we analyze 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 13 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) as shown in Fig. 16. It means that a high-order Markov chain works well for sequential recommendation in our proposed method.

4.6.2 Handling sparsity

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

Variants for baby and garden dataset were also created in the same way. Models' performance was evaluated using same parameters as explained in Sect. 4.2. The results are presented in Fig. 17 which shows that SEMMRec outperforms the other approaches on different sizes and the length of sequences of datasets. In particular, in small size dataset 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



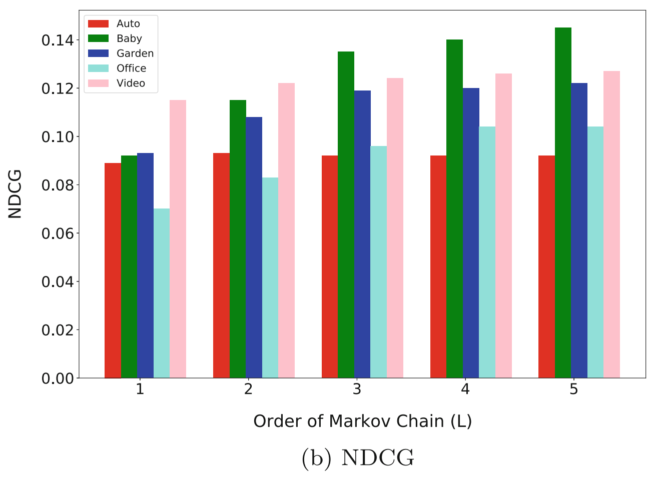


Fig. 16 Effects of different order of L on proposed model (SEMMRec) performance based on **a** recall and **b** NDCG

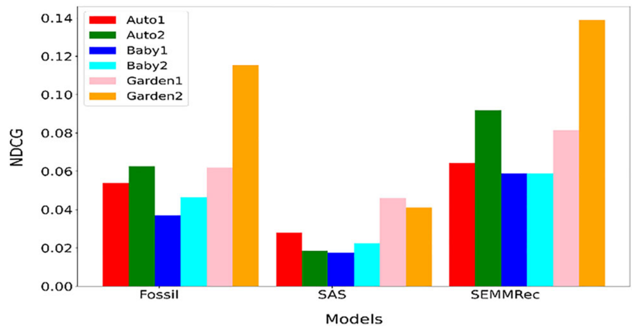


Fig. 17 Performance comparison of proposed model using sparse datasets (two variants of each dataset)

that our proposed approach is mainly stable on various sparse datasets (with different size and sequence).

4.6.3 Influence of train and test split

For dataset 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). In the temporal user splitting, we used train and test splits of (a) 70%, 30% and (b) 80%, 20%. Availability of a rating (Amazon) is considered as user-item interaction and we used timestamps to determine the sequential order of actions. Data were 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 were 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. Users with greater than five 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' metadata (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. Prediction performance of the proposed model with different train and test split strategies is shown in Table 14.

Table 14 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

Declarations

Conflict of interest On behalf of all authors, the corresponding author stat-es that there is no conflict of interest.

References

1. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_7
2. Aggarwal, C.C.: An introduction to recommender systems. In: Aggarwal, C.C. (ed.) Recommender Systems: The Textbook, pp. 1–28. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29659-3_1
3. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14 (1995). https://doi.org/10.1109/ICDE.1995.380415
4. Asher, R.E., Simpson, J.M.Y.: The Encyclopedia of Language and Linguistics. Pergamon (1993)
5. Ayres, J., Flannick, J., Gehrke, J., 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, pp. 429–435. Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/775047.775109
6. Bernhard, S.D., Leung, C.K., Reimer, V.J., Westlake, J.: Click-stream Prediction Using Sequential Stream Mining Techniques with Markov Chains. In: Proceedings of the 20th International Database Engineering & Applications Symposium, IDEAS '16, pp. 24–33. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2938503.2938535
7. Bhatta, R., Ezeife, C.I., 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., Khalil, I. (eds.) Big Data Analytics and Knowledge Discovery. Lecture Notes in Computer Science, pp. 57–72. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27520-4_5
8. Blum, A., Hopcroft, J., Kannan, R.: Foundations of Data Science, 1st edn. Cambridge University Press, Cambridge (2020). https://doi.org/10.1017/9781108755528
9. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl. Based Syst. **46**, 109–132 (2013). https://doi.org/10.1016/j.knosys.2013.03.012.
10. Brafman, R.I., Heckerman, D., Shani, G.: Recommendation as a Stochastic Sequential Decision Problem p. 10
11. Cao, L.: Actionable knowledge discovery and delivery. Wiley Interdiscipl. Rev. Data Min. Knowl. Discov. **2**(2), 149–163 (2012)
12. Cao, L.: Combined mining: analyzing object and pattern relations for discovering and constructing complex yet actionable patterns. Wiley Interdiscipl. Rev. Data Min. Knowl. Discov. **3**(2), 140–155 (2013)
13. Cao, L., Zhang, C.: Domain-driven data mining: a practical methodology. Int. J. Data Warehous. Min. (IJDWM) **2**(4), 49–65 (2006)
14. Cao, L., Zhang, C., Zhao, Y., Yu, P.S., Williams, G.: DDDM2007: domain driven data mining. ACM SIGKDD Explor. Newslett. **9**(2), 84–86 (2007). https://doi.org/10.1145/1345448.1345467
15. Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., Park, E.K.: Flexible frameworks for actionable knowledge discovery. IEEE Trans. Knowl. Data Eng. **22**(9), 1299–1312 (2009)
16. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential Recommendation with User Memory Networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, Marina Del Rey CA USA, pp. 108–116 (2018). https://doi.org/10.1145/3159652.3159668
17. Deshpande, M., Karypis, G.: Selective Markov models for predicting web page accesses. ACM Trans. Internet Technol. **4**(2), 163–184 (2004)
18. de Gemmis, M., Lops, P., Musto, C., Narducci, F., Semeraro, G.: Semantics-aware content-based recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 119–159. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_4
19. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative Filtering Recommender Systems. Now Publishers Inc, Norwell (2011)
20. Faridani, V., Jalali, M., Jahan, M.V.: Collaborative filtering-based recommender systems by effective trust. Int. J. Data Sci. Anal. **3**(4), 297–307 (2017). https://doi.org/10.1007/s41060-017-0049-y
21. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Mag. **17**(3), 37 (1996)
22. Fayyad, U.M., Piatetsky-Shapiro, G., Uthurusamy, R.: Summary from the kdd-03 panel: data mining: the next 10 years. ACM Sigkdd Explor. Newslett. **5**(2), 191–196 (2003)
23. Gabrilovich, E., Markovitch, S.: Wikipedia-based semantic interpretation for natural language processing. J. Artif. Intell. Res. **34**, 443–498 (2009). https://doi.org/10.1613/jair.2669.
24. Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., 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, pp. 1809–1818. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2783258.2788627
25. Guo, W., Wang, S., Lu, W., Wu, H., Zhang, Q., 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), pp. 1–10. IEEE (2021)
26. Harris, Z.S.: Distributional structure. WORD **10**(2–3), 146–162 (1954). https://doi.org/10.1080/00437956.1954.11659520
27. He, R., McAuley, J.: Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, Barcelona, Spain, pp. 191–200 (2016). https://doi.org/10.1109/ICDM.2016.0030. http://ieeexplore.ieee.org/document/7837843/
28. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939 [cs] (2016)
29. Hidasi, B., Quadrana, M., Karatzoglou, A., 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, pp. 241–248. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2959100.2959167
30. Ishigaki, T., Terui, N., Sato, T., Allenby, G.M.: Personalized market response analysis for a wide variety of products from sparse transaction data. Int. J. Data Sci. Anal. **5**(4), 233–248 (2018). https://doi.org/10.1007/s41060-018-0099-9
31. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010). Google-Books-ID: eygTJBd_U2cC
32. Kabbur, S., Ning, X., 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, pp. 659–667. ACM, Chicago Illinois USA (2013). https://doi.org/10.1145/2487575.2487589
33. Kang, W.C., McAuley, J.: Self-Attentive Sequential Recommendation. arXiv:1808.09781 [cs] (2018)

34. Karatzoglou, A., Baltrunas, L., Shi, Y.: Learning to rank for recommender systems. In: Proceedings of the 7th ACM conference on Recommender systems, pp. 493–494. ACM, Hong Kong China (2013). https://doi.org/10.1145/2507157.2508063
35. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 447–456. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1557019.1557072
36. Kumara Swamy, M., Krishna Reddy, P.: A model of concept hierarchy-based diverse patterns with applications to recommender system. Int. J. Data Sci. Anal. **10**(2), 177–191 (2020). https://doi.org/10.1007/s41060-019-00203-2
37. Kiran, R., Kumar, P., Bhasker, B.: DNNRec: a novel deep learning based hybrid recommender system. Expert Syst. Appl. **144**, 113054 (2020)
38. Li, J., Li, L., Wu, Y., Chen, S.: An improved recommender based on hidden Markov model p. 5
39. Li, T., Choi, M., Fu, K., Lin, L.: Music Sequence Prediction with Mixture Hidden Markov Models. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 6128–6132 (2019). https://doi.org/10.1109/BigData47090.2019.9005695
40. Liu, T., Wang, Z., Tang, J., Yang, S., Huang, G.Y., Liu, Z.: Recommender systems with heterogeneous side information. arXiv:1907.08679 [cs, stat] (2019)
41. Mabroukeh, N.R., Ezeife, C.I.: A taxonomy of sequential pattern mining algorithms. ACM Comput. Surv. **43**(1), 3:1-3:41 (2010). https://doi.org/10.1145/1824795.1824798
42. Middleton, S.E., Shadbolt, N.R., De Roure, D.C.: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst. **22**(1), 54–88 (2004). https://doi.org/10.1145/963770.963773
43. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013). http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
44. Nasir, M., 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), pp. 270–274 (2020). https://doi.org/10.1109/ASONAM49781.2020.9381352. ISSN: 2473-991X
45. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543.
46. Polyzou, A., Karypis, G.: Scholars walk: a Markov chain framework for course recommendation p. 6 (2019)
47. Quadrana, M., Cremonesi, P., Jannach, D.: Sequence-aware recommender systems. arXiv:1802.08452 [cs] (2018)
48. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pp. 130–137. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3109859.3109896
49. Raza, S., Ding, C.: A regularized model to trade-off between accuracy and diversity in a news recommender system. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 551–560. IEEE, Atlanta, GA, USA (2020). https://doi.org/10.1109/BigData50022.2020.9378340. https://ieeexplore.ieee.org/document/9378340/
50. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 811–820. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1772690.1772773
51. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 1–34. Springer, Boston (2015). https://doi.org/10.1007/978-1-4899-7637-6_1
52. Rubino, G., Sericola, B.: Markov Chains and Dependability Theory. Cambridge University Press, Cambridge (2014). https://doi.org/10.1017/CBO9781139051705
53. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Series in Computer Science. Addison-Wesley, Reading (1988)
54. Sarukkai, R.R.: Link prediction and path analysis using Markov chains. Comput. Netw. **33**(1–6), 377–386 (2000). https://doi.org/10.1016/S1389-1286(00)00044-X.
55. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, pp. 291–324. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-72079-9_9
56. Schafer, J.B., Konstan, J.A., Riedl, J.: E-Commerce recommendation applications. Data Min. Knowl. Discov. **5**(1), 115–153 (2001). https://doi.org/10.1023/A:1009804230409
57. Semeraro, G., Degemmis, M., Lops, P., 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, pp. 2856–2861. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007)
58. Semeraro, G., Lops, P., Basile, P., de Gemmis, M.: Knowledge infusion into content-based recommender systems. In: Proceedings of the 3rd ACM Conference on Recommender Systems, RecSys '09, pp. 301–304. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1639714.1639773
59. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. J. Mach. Learn. Res. **6**(Sep), 1265–1295 (2005). https://www.jmlr.org/papers/v6/shani05a
60. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009**, 1–19 (2009). https://doi.org/10.1155/2009/421425.
61. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1441–1450. ACM, Beijing China (2019). https://doi.org/10.1145/3357384.3357895
62. Tang, J., Wang, K.: Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. arXiv:1809.07426 [cs] (2018)
63. Turney, P.D., Pantel, P.: From frequency to meaning: vector space models of semantics. J. Artif. Intell. Res. **37**, 141–188 (2010). https://doi.org/10.1613/jair.2934.
64. Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., Lee, D.L.: Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 839–848. ACM, London United Kingdom (2018). https://doi.org/10.1145/3219819.3219869
65. Wang, X., Kadıoğlu, S.: Modeling uncertainty to improve personalized recommendations via Bayesian deep learning. Int. J. Data Sci. Anal. (2021). https://doi.org/10.1007/s41060-020-00241-1
66. Xiao, Y., Ezeife, C.I.: E-Commerce product recommendation using historical purchases and clickstream data. In: Ordonez, C., Bellatreche, L. (eds.) Big Data Analytics and Knowledge Discovery.

Lecture Notes in Computer Science, pp. 70–82. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98539-8_6

67. Xiao, Y., Yao, L., Pei, Q., Wang, X., Yang, J., Sheng, Q.Z.: Mgnn: mutualistic graph neural network for joint friend and item recommendation. IEEE Intell. Syst. **35**(5), 7–17 (2020)
68. Zhang, C., Philip, S.Y., Bell, D.: Introduction to the domain-drive data mining special section. IEEE Trans. Knowl. Data Eng. **22**(6), 753–754 (2010)
69. Zhao, H., Yao, Q., Song, Y., Kwok, J., Lee, D.L.: Side information fusion for recommender systems over heterogeneous information network. arXiv:1801.02411 [cs] (2020)
70. Zhong, M., Li, C., Wen, J., Liu, L., Ma, J., Zhang, G., Yang, Y.: Hignet: hierarchical and interactive gate networks for item recommendation. IEEE Intell. Syst. **35**(5), 50–61 (2020)
71. Zimdars, A., Chickering, D.M., Meek, C.: Using temporal data for making recommendations. arXiv:1301.2320 [cs] (2013)