ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ ПРОБЛЕМИ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ ПІДТРИМАННЯ ПРИЙНЯТТЯ РІШЕНЬ

UDC 004.85
DOI: 10.20535/SRIT.2308-8893.2025.1.01

# THE HYBRID SEQUENTIAL RECOMMENDER SYSTEM SYNTHESIS METHOD BASED ON ATTENTION MECHANISM WITH AUTOMATIC KNOWLEDGE GRAPH CONSTRUCTION

**D.V. ANDROSOV, N.I. NEDASHKOVSKAYA**

**Abstract.** Sequential personalized recommendations, such as next best offer prediction or modeling demand evolution for next basket prediction, remain a key challenge for businesses. In recent years, deep learning models have been applied to solve these problems and demonstrated high feasibility. With the introduction of graph-based deep learning, it has become easier to perform collaborative filtering and link prediction tasks. The current paper proposes a new method of building a recommender system using a graph representation learning framework in combination with deep neural networks for sequence-to-sequence modeling and statistical learning for sequence-to-graph mapping. Benchmarking model performance on an online retail store visits dataset provides evidence of the method's ranking capabilities.

**Keywords:** recommender system, graph neural network, graph embeddings.

## INTRODUCTION

First, it is necessary to coin the term "recommender system" to reach a common understanding of this notion throughout the following work. A recommendation should reflect the strong relationship between user activities and item relations [1]. As an example, a user's preference for a historical documentary is highly correlated with dedicating more time to watch another documentary or other educational program, rather than an action film [1]. These types of relations can be set explicitly by the group of experts on a level of high granularity or determined in a data-driven way on an item level.

A recommender system (RS) is a set of statistical models that considers the certain user's interaction history, knowledge about this user and each item from the available interactions, and provides relevant content (recommendation) [2]. Relevancy means an ordering relation of the user's likelihood to interact with the set of available items. Hence there exists a broad range of recommender approaches, such as non-personalized, semi-personalized, and personalized [2]. In the scope of current work, we focus on the development of the personalized RS, thus terms "recommender system" and "personalized recommender system" along with their abbreviations are interchangeable.

In recent years, personalized RS have achieved significant success across various real-world applications, including e-commerce platforms, streaming media, and online retail industries. A particularly notable area of RS application is the next best offer (NBO) recommendation task, which involves predicting which item a user will view or purchase after interacting with the platform.

NBO, also known as next best action (NBA) [3], or generalized as next-basket recommendation (NBR) [4], is a prevalent use case for any enterprise engaged in business-to-consumer (B2C) operations. Marketing teams in such enterprises have been implementing NBO/NBA projects for many years. However, many of these projects fail to deliver the expected returns [3]. This lack of performance can be caused by several factors: the use of traditional methods, failure to retrain NBO models with new sets of features (resulting in underutilization of both breadth and depth of available data), lack of effective campaign validation methods, technological deficiencies, etc.

The advent of machine learning has provided a renewed perspective on NBO/NBR. There is now an opportunity to utilize these technologies, along with comprehensive data, to enhance and optimize basket recommendations more effectively than before.

As an example, by leveraging deep learning approaches, the delivery of personalized offers and recommendations was significantly enhanced, overall demonstrating improvements in customer engagement. These enhancements can lead to increased customer satisfaction and loyalty, which ultimately drive higher sales and revenue for the business [4; 5].

## RELATED WORK

The sequential recommendation task and the next best action task aim to produce such a recommended item set that satisfies the condition of relevance given the most recent user interactions. More strictly, given user $u \in U$, user session vector $s \in S$, where $S$ is the collection of subsets of item set $S$, candidate item set $I \in I_d$, and the content-dependent relevancy function $R: U \times S \times I_d \to [0,1]$, $d \in \mathrm{N}$, one can formulate the sequential recommendation task as:

$$I^{*d} = \arg\max R(u, s, I^d),$$

where $R$ could be any real-valued function, but usually the likelihood function

$$R(u, s, I^d) = p(u, s \mid I^d(\theta))$$

is applied [6].

Since user sessions could be of arbitrary length, the objective of capturing long-term dependencies between session items and candidate ones is a key one. To accomplish this task, models based on high-order Markov chains were proposed, such as context tree models (CT) [6; 7] and Markov chain similarity models [8].

Context trees construct a partition tree for each user session and then define a high-order Markov chain by traversing this tree [7].

As an alternative, the combination of high-order Markov chains with similarity-based methods, such as sparse linear methods (SLIM) and factored item similarity models (FISM) allows capturing short-term and long-term user-item and item-item relations simultaneously [8].

Besides Markov chain models, deep learning models are widely used for solving sequential recommendation tasks. Among all deep learning architectures, recurrent neural networks (RNN) are most widely chosen for accomplishing the sequential recommendation problem, especially long short-term memory networks (LSTM), and their modifications, such as bi-directional LSTM [9; 10]. Similarly to Markov chain-based hybrid approaches, LSTM networks are often used to capture session-level patterns. Thus, to enrich the proposed recommendations with learned long-term behavioural patterns, rule-based recommender design is applied [9]. Alternatively, bidirectional LSTM model was leveraged to infer recommendation rules based on current and previous user sessions [10].

Recently, a relatively new family of neural networks was applied for this task, called attention networks [11]. Attention networks are ubiquitously applied in recommendation retrieval tasks, e.g. hierarchical attention networks, that considers inputs of user-item and item-item interactions to predict further user actions [12] or stochastic self-attention networks to produce next recommendation candidates [13].

## MATERIALS AND METHODOLOGY

For solving the sequential recommendation retrieval task, it is proposed to rely on graph neural network (GNN) framework.

GNNs are a family of deep neural networks capable to perform inference on data structured as graphs [14; 15; 17]. During the training process each vertex $v \in V$ of graph $G(V, E)$ updates its representation by aggregating features from its neighbors. This process can be formalized as:

$$h_v^{(k+1)} = \sigma\left( \sum_{u \in N(v)} W^{(k)} h_u^{(k)} + b^{(k)} \right),$$

where $h_v^{(k)}$ is the feature vector of vertex $v$ at layer $k$, $N(v)$ represents the neighbors of $v$, $W^{(k)}$ and $b^{(k)}$ are the layer-specific weights and biases, and $\sigma$ is a non-linear activation function.

After choosing the framework, it is necessary to somehow interpret the input sequences for the network in form of a graph.

It is intuitively understood, that user-item, user-user and item-item relations could be naturally represented as a graph $G = G(V, E, w, f)$, where $V = \langle U, I \rangle$ is a set of users and items (i.e. vertices of a graph), $E = ((u,i) \mid u \in U, i \in I)$ is an edge set, and $f : E \to w$ is a mapping from edge set to edge weights $w \in \mathrm{R}$. Thus, geometrical deep learning frameworks, such as graph neural networks (GNN) are applied to solve recommendation retrieval tasks [16].

Embedding of such graph, i.e. dense vector collection of node relations could be obtained by processing $G$ through graph convolution network (GCN) [15] and attention network [14]. As an example, [14] combines attention mechanism with graph convolutional networks [15] to capture embeddings of user-item graph and produce next item recommendation for a given user.

GCN updates node features by aggregating features from neighboring nodes, using the following formula [15]:

$$h^{(k+1)} = \sigma\left( \widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2} h^{(k)} W^{(k)} \right),$$

where $\widetilde{A} = A + I$ is the adjacency matrix $A$ with added self-loops (identity matrix $I$); $\widetilde{D}$ is the degree matrix corresponding to $\widetilde{A}$; $h^{(k)}$ represents the node features at layer *k*; $W^{(k)}$ is the weight matrix at layer *k*; $\sigma$ is an activation function.

By further considering the assumption that user preferences are non-static in long-term perspective, it is proposed to add to graph *G* (continuous) time dimension $t \in \mathbb{R}$ and obtain in such way dynamic graph $G^{(t)}(V^{(t)}, E^{(t)}, w^{(t)}, f)$. In such case, every user session at time *t* is generated by traversing the graph $G^{(t)}$. However, synthesis of such graph at a step *t* = 0 is required based on previous interactions.

For solving the task above, it is proposed to use rule mining algorithms, such as computing pointwise mutual information (PMI) [18] between each pair of bought/seen items, FP-Growth trees [19] and TF-IDF metric [20] for selecting the most relevant item combinations in the context of each session.

**Pointwise mutual information for knowledge graph synthesis**

Consider a set of user sessions/transactions $D = \{s_1, s_2, ..., s_m\}$, where each transaction $s_j$ is a set of items $\{i_1, i_2, ..., i_k\}$ from a larger set of items $i$.

Let $p(i)$ denote the probability of item *i* occurring in a transaction, and $p(i, j)$ denote the joint probability of items *i* and *j* co-occurring in the same transaction. These probabilities can be estimated as follows:

$$p(i) = \frac{|D_i|}{|D|}; \quad p(i, j) = \frac{|D_{i,j}|}{|D|},$$

where $|D_i|$ is the number of transactions containing item *i*, and $|D_{i,j}|$ is the number of transactions containing both *i* and *j*.

The PMI between two items *i* and *j* is calculated using the formula:

$$PMI(i, j) = \log \frac{p(i, j)}{p(i)p(j)}.$$

Then by introducing the threshold $\varepsilon$ and indicator function $\mathbf{1}_{PMI(i,j)\geq\varepsilon}$, one can construct a knowledge graph $G^{(0)} = G(V, E, \mathbf{1}_{PMI(i,j)\geq\varepsilon}, f)$.

**FP-Growth algorithm for rule mining and knowledge graph synthesis**

The FP-Growth (Frequent Pattern Growth) algorithm is a widely used method for mining frequent itemsets in large datasets, particularly in the context of association rule learning. The FP-Growth algorithm is more efficient and scalable compared to other rule mining algorithms like Apriori, since it does not explicitly generate itemsets, but builds FP-Growth tree for this purpose [19].

Consider a set of user sessions $D = \{s_1, s_2, ..., s_m\}$, the FP-tree *G* is defined as:

$$G = \{g \mid g = (Name(g), Supp(g), Child(g))\},$$

where each node $g$ is a structure storing the node's name (*Name*($g$)), its support value (*Supp*($g$)), and a set of references to its child nodes (*Child*($g$)). Items $i \in I$ are vertices of the FP-tree. The path from the root $g_0$ to a vertex $g$ represents a set of items $F \subseteq I$. Let $G(i) = \{g \in G \mid g = i\}$ be the set of vertices corresponding to item $i \in I$. The support of item $i$ is calculated as follows:

$$Supp(i) = \sum_{g \in G(i)} Supp(g).$$

To construct the FP-tree, items $i \in I$ are ordered in descending order of their support *Supp*($i$). Items with support below the minimum threshold $\text{Supp}_{\min}$ are excluded from further consideration. For each element in each sorted transaction in the initial session/transaction database (TDB), the FP-tree nodes are constructed as follows:

- If a descendant of the current node exists that contains the current element, no new node is created, and the support of this descendant is incremented by one.
- Otherwise, a new descendant node is created with support initialized to one.
- The newly found or created node becomes the current node.

Thus, the levels of the FP-tree correspond to items ordered by descending support values *Supp*($i$), resulting in a specific order for the set of items.

For FP-Growth trees, it is proposed to clip the maximum height of tree at 2, thus limiting the maximum frequent pattern length to be 2. Then, the graph $G^{(0)}$ is directly obtained from the frequent patterns of lengths 1 and 2, and the weight mapping is defined as follows:

$$w : e \to Supp(e).$$

**Leveraging TF-IDF for knowledge graph construction**

Term Frequency-Inverse Document Frequency (TF-IDF) is a metric that is commonly used to evaluate the importance of a word in a document from a collection of documents, called corpus. It is widely used in text mining and information retrieval to identify significant words in documents [20]. The term frequency TF(*t,d*) is the number of times a term $t$ appears in a document $d$. It is often normalized to prevent bias towards longer documents:

$$TF(t,d) = \frac{\left|\{t : t \in d\}\right|}{\left|\{\hat{t} : \hat{t} \in d\}\right|},$$

where $\left|\{t : t \in d\}\right|$ is the number of times term $t$ appears in document $d$, and $\left|\{\hat{t} : \hat{t} \in d\}\right|$ is the total number of terms in document $d$. The inverse document frequency IDF(*t,D*) measures how important a term is across the corpus:

$$IDF(t,D) = \log \frac{\left|D\right|}{\left|\{d \in D : t \in d\}\right|},$$

where $\left|\{d \in D : t \in d\}\right|$ is the quantity of documents that contain the term $t$. Note that for the $\text{IDF}(t,D)$ it is encouraged to use logarithm of inverse-document frequency, hence rare items will not receive too big values of TF-IDF score.

After obtaining the latter measures, the TF-IDF score for a term $t$ in a document $d$ is retrieved by multiplying the latter quantities:

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) .$$

It is proposed to apply the TF-IDF metric to each pair of user sessions items $(i,j) \in s$ and by interpreting the tuple $(i,j)$ as a single term in document $u$. After the following operation, by introducing the threshold $\varepsilon$ and indicator function $\mathbf{1}_{TF-IDF(i,j)\geq\varepsilon}$, one can construct a knowledge graph $G^{(0)} = G(V,E,\mathbf{1}_{PMI(i,j)\geq\varepsilon},f)$.

**Graph Neural Network training algorithm**

After obtaining such graph $G^{(0)}$, it is proposed to create a deep neural network, which algorithm of fitting is shown on Fig. 1.

**Algorithm 1** Recommendation retrieval network fitting algorithm (single forward and update pass)

**Require:** $u$ - user vector, $S_u$ - set of sessions, $s^*$ - next best offer item for session $s$, Adj $\left(G^{(0)}\right)$ - adjacency matrix of a knowledge graph $G^{(0)}$, $T$ - discrete time, $\alpha \in [0,1]$ - neighbor decay factor, $E_I$ - item embedding matrix, $E_U$ - users embedding matrix;

Adj$_0$ := Adj $\left(G^{(0)}\right)$
$e_u := E_U(u)$
**for** $t = 0,\ldots,T$
  **for all** $s \in S_u$
    $e_{true} := E(s^*)$
    $A := \varnothing$
    **for all** $i \in s$
      calculate sum of embeddings of all neighbors of the item $i$ from the graph $G^{(t)}$ and call it $\Sigma_i$;
      $\Sigma_i := (1-\alpha)E_I(i) + \alpha\Sigma_i$;
      calculate self-attention over $\Sigma_i$ and call it $a_i$;
      $A \longleftarrow a_i$
    pass the obtained matrix $A$ to the multilayer perceptron and obtain the vector $e_{predicted}$;
    Calculate the loss between $e_{predicted}$ and $e_{true}$
  Update Adj$_t$, $E_I$ and $E_U$ according to backpropagation algorithm

*Fig. 1.* GNN training algorithm

The loss function in the context of the proposed algorithm is the cross-entropy function:

$$H(e_{predicted}, e_{true}) = E_{e_{predicted}}(e_{true}) ,$$

where $E_p(q)$ is the expected value of random variable $q$ with respect to the random variable $p$.

For the top-$k$ retrieval phase for the given user $u$ and its session $s$, it is proposed to calculate affinity scores between embedding each item $i \in I$ and retrieved embedding representation of the pair $\langle u,s \rangle$. In the scope of the current approach, it is proposed to use the cosine similarity function:

$$\cos(i,j) = \frac{i^{\mathrm{T}} j}{\|i\| \cdot \|j\|} .$$

**Warm embedding initialization**

Since the proposed GNN operates with the adjacency matrix of a knowledge graph $G^{(t)}$ and user and item properties' embeddings $E_U$ an $E_I$, respectively, it

is crucial to initialize such embedding matrices in a way to preserve the relationship between embeddings, depicted in the original graph. Thus, it is recommended to leverage warm embedding initialization technique before starting model training.

Warm embedding initialization is the practice of initializing the embeddings with pre-trained values rather than random values. This technique leverages prior knowledge to potentially enhance the model's performance, particularly during the initial stages of training.

One can pre-compute embeddings by using such natural language processing (NLP) models as Word2Vec [21], GloVe [22], and FastText [23].

Since it is necessary to depict the relationships formulated in knowledge graph $G^{(0)}$, it is proposed to pre-compute embeddings for $G^{(0)}$ using Node2Vec algorithm [24].

The main idea is to generate random walks on the graph and treat these walks as sentences, where each node corresponds to a lexeme, e.g. a word or item identifier.

Node2Vec generates biased random walks starting from each node to explore the graph. The algorithm introduces two parameters, $p$ and $q$:

- $p$ is a return parameter, which controls the likelihood of revisiting a node in the walk. A higher value makes it less likely to revisit the previous node, promoting exploration.
- $q$ is an in-out parameter, which controls whether the likelihood of visiting nodes is close or far from the starting node. A higher value biases the walk towards breadth-first search, while a lower value biases it towards depth-first search.

Once the random walks are generated, Node2Vec utilizes the same Skip-Gram approach of learning lexeme embedding, as in the original Word2Vec architecture [24]. The objective is to maximize the following probability of observing a node's neighborhood given its embedding:

$$\max \sum_{u \in V} \sum_{v \in N(u)} \log p(v \mid u),$$

where $V$ is the set of nodes, $N(u)$ is the neighborhood of node $u$, and $p(v \mid u)$ is the probability of node $v$ being a neighbor of node $u$, modeled using the embeddings.

## EXPERIMENTS AND RESULTS

### Dataset description

For the experiment purposes, the dataset was selected from an e-commerce consumer electronics retail platform. The schema of the data is presented in the Table 1.

**Table 1.** Dataset fields description

| Column name | Column type | Column description |
|---|---|---|
| user_id | unsigned bigint | user identifier |
| product_id | unsigned bigint | product identifier |
| category_id | unsigned bigint | item category identifier |
| event_type | string | interaction type (view, purchase) |
| user_session | base64 | user session identifier |

The dataset for evaluation of proposed method is a two-month snapshot from an anonymous e-commerce store database and has approximately 69 million records and 6.5 million sessions (9 GB of raw data).

**Models' configurations**

For the sake of qualitative assessment of experiment results, i.e. understanding the potential benefits of leveraging the proposed approach, it was decided to use a LSTM-based deep neural network as a baseline model for sequential recommendation.

For the experiment purposes, the following models' configuration was selected after performing hyperparameters tuning:

- embedding dimension size – 64;
- number of multi-layer perceptron layers – 2;
- prediction per session – 1 item, i.e. next best offer prediction;
- neighbor decay parameter $\alpha$ – 0.5;
- training step – 0.01;
- number of epochs – 20.

For Node2Vec algorithm, the following configuration was set:

- embedding dimension size – 64;
- number of random walk stages – 3;
- window size for skip-grams generation – 10;
- number of negative samples per skip-gram sequence – 5;
- training step – 0.01;
- number of epochs – 5.

For automated knowledge graph generation for different approaches, the given thresholds for minimum values were selected:

- for FP-Growth algorithm, the minimum support rate was set at 0.05% (which roughly correspond to be 1 in 3450 sessions);
- for TF-IDF filtering, the threshold for TF-IDF score for each item was set at 0.3;
- for PMI-based algorithm, the minimum PMI was set at the value of 12.

In order to evaluate the quality of recommendation retrieval, the most popular ranking metrics Mean Average Precision@$k$ and Negative Discounted Cumulative Gain@$k$ were chosen along with average training time per epoch in seconds.

**Results and discussions**

The results of model evaluation over the dataset are shown in Table 2.

**Table 2.** Model evaluation

| Model | Avg. time per epoch, s | MAP@1 | MAP@10 | NDCG@1 | NDCG@10 |
|---|---|---|---|---|---|
| baseline LSTM | 0.2670 | 0.0389 | 0.0937 | 0.0388 | 0.1217 |
| proposed GNN w. PMI | 610.02 | 0.1442 | 0.2551 | 0.1444 | 0.2988 |
| proposed GNN w. TF-IDF | 115.08 | 0.0844 | 0.1622 | 0.0845 | 0.1960 |
| proposed GNN w. FP-Growth | 53.920 | 0.0399 | 0.0887 | 0.0378 | 0.1150 |

Overall, GNNs performs significantly better than baseline LSTM, despite their training time being slower more than 200 times. Such slowdown is caused by traversing each time the synthesized knowledge graph $G^{(t)}$. From the comparison above one can decide to stick either with TF-IDF option which provides the compromise in terms of both accuracy and learning time, or experimenting with PMI clipping for achieving the 4x increase in recommendation retrieval accuracy in the scope of next best action prediction. Using FP-Growth algorithm for automatic knowledge graph construction wasn't beneficial nor in accuracy of predictions, offering the same results as the baseline LSTM model, but yet being much slower.

The evolution of the MAP@1 metric for all models is depicted on Fig. 2. The evolution of the MAP@10 metric for all models is depicted on Fig. 3.

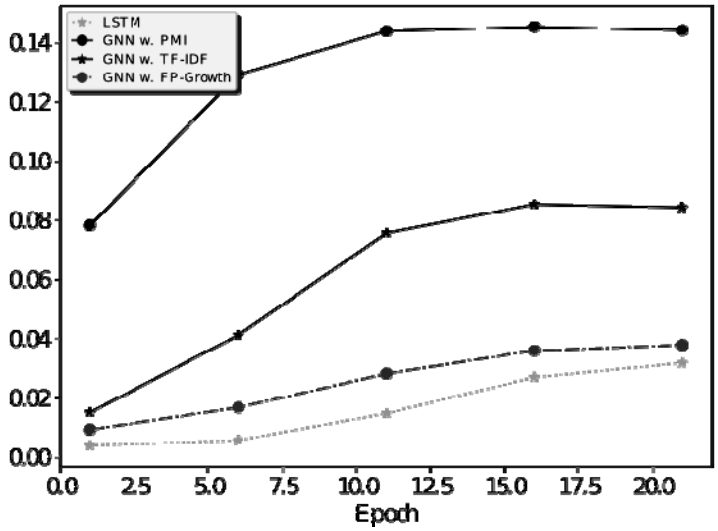


*Fig. 2.* MAP@1 per epoch

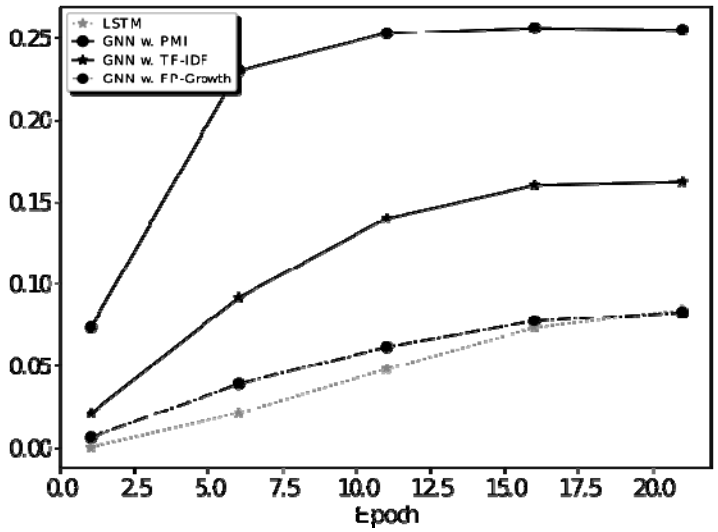


*Fig. 3.* MAP@10 per epoch

On the other hand, the evolution of the NDCG@1 and NDCG@10 metric for all models are depicted on Fig. 4 and Fig. 5, respectively.

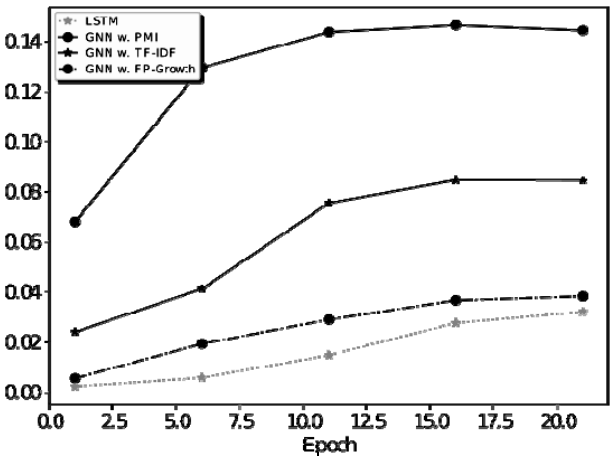


*Fig. 4.* NDCG@1 per epoch

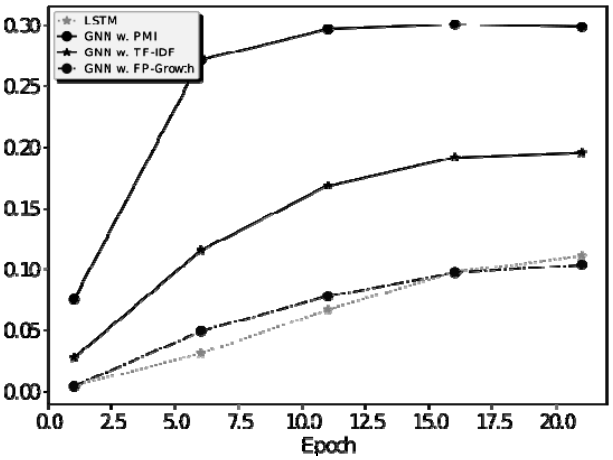


*Fig. 5.* NDCG@10 per epoch

By examining the results one can conclude that the proposed approach is feasible to use in the context of the next-best-offer recommendation.

## CONCLUSIONS

In the current research the problem of personalized sequential recommendations was addressed. By conducting a literature review on a given topic, it was discovered that by leveraging hybrid approaches, including Context Trees and high-order Markov chains, the problem of sequential recommendation was solved, but with a severe limitation — order of items to consider depends on order of Markov chain.

By introduction of deep learning and leveraging the ability of (artificial) neural networks to be a universal approximators, to solve the given problem, deep recurrent neural networks, namely LSTMs were used, since of their ability of capturing temporal dependencies of arbitrary lengths.

However, RNNs are requiring the data to be only in the form of a sequence, thus discarding the ability to naturally represent user-user, item-item and user-item relations as a network. Such a way of representing these relations and capturing them as a knowledge graph, using data mining algorithms, is a key idea of the proposed research.

It was also proposed by the current paper to slightly change the mechanics of an Attention mechanism, to allow capture not only the studied sequence, but examine associations, presented in a knowledge graph, for each item in the sequence.

By examining different techniques to capture associative similarities between sets of items, namely TF-IDF, FP-Growth trees and PMI metric, constructing a deep graph neural network with proposed modification of Attention mechanism, and leveraging Node2Vec algorithm to retrieve the embedding matrix of a knowledge graph, new method for hybrid sequential recommender system design was introduced.

By benchmarking the models on an anonymous retailer dataset, the evidence of high precision of GNN models in terms of MAP@*k* and NDCG@*k* for next-best offer recommendation was obtained.

## REFERENCES


1. C. Aggarwal, *Recommender Systems*: The Textbook. Springer, 2016.
2. K. Falk, *Practical Recommender Systems*. Manning, 2019.
3. A. Rasool, *Next Best Offer (NBO) / Next Best Action (NBA)- why it requires a fresh perspective?* 2019. [Online]. Available: https://www.linkedin.com/pulse/next-best-offer-nbo-action-nba-why-requires-fresh-azaz-rasool/
4. S. Wang et al., "Modeling User Demand Evolution for Next-Basket Prediction," IEEE Transactions on Knowledge and Data Engineering, vol. 35(11), pp. 11585–11598, 2023. doi: https://doi.org/10.1109/TKDE.2022.3231018
5. K.A. Eliyahu, *Achieving Commercial Excellence through Next Best Offer models*. [Online]. Available: https://www.linkedin.com/pulse/achieving-commercial-excellence-through-next-best-offer-kisliuk/
6. S. Wang et al., "Sequential Recommender Systems: Challenges, Progress and Prospects," in *IJCAI 2019*, 2019. doi: https://doi.org/10.24963/ijcai.2019/883
7. F. Garcin, C. Dimitrakakis, and B. Faltings, "Personalized News Recommendation with Context Trees," in *Proceedings of the 7th ACM conference on Recommender systems, ACM, 2013*. doi: https://doi.org/10.1145/2507157.2507166
8. R. He, J. McAuley, "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation," in 2016 *IEEE 16th International Conference on Data Mining (ICDM), Barcelona, 2016*, pp. 191–200. doi: 10.1109/ICDM.2016.0030
9. Q. Xia et al., "Modeling Consumer Buying Decision for Recommendation Based on Multi-Task Deep Learning," in *CIKM '18*, 2018. doi: https://doi.org/10.1145/3269206.3269285
10. C. Zhao, J. You, X. Wen, and X. Li, "Deep Bi-LSTM Networks for Sequential Recommendation," *Entropy* (*Basel*), 22(8), 870, 2020. doi: https://doi.org/10.3390/e22080870
11. A. Vaswani et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NIPS 2017)*, pp. 5998–6008, 2017. doi: https://doi.org/10.48550/arXiv.1706.03762
12. H. Ying et al., "Sequential Recommender System based on Hierarchical Attention Network," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, 13–19 July 2018*, pp. 3926–3932. doi: https://doi.org/10.24963/ijcai.2018/546
13. Z. Fan et al., "Sequential Recommendation via Stochastic Self-Attention," in *Proceedings of the ACM Web Conference 2022 (WWW '22), 2022*. doi: https://doi.org/10.1145/3485447.3512077
14. T. Hekmatfar, S. Haratizadeh, P. Razban, and S. Goliaei, "Attention-Based Recommendation On Graphs," *arXiv*. 2022. [Online]. Available: https://arxiv.org/abs/2201.05499
15. T.N. Kipf, M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.02907
16. N.I. Nedashkovskaya, D.V Androsov, "Multicriteria evaluation of recommender systems using fuzzy analytic hierarchy process," PREPRINT (Version 1), available at *Research Square*. doi: https://doi.org/10.21203/rs.3.rs-3228086/v1

17. Z. Wu et al., "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), pp. 4–24, 2020. doi: https://doi.org/10.48550/arXiv.1901.00596
18. K.W. Church, P. Hanks, "Word association norms, mutual information, and lexicography," *Comput. Linguist*, vol. 16(1), pp. 22–29, 1990.
19. J. Han, J. Pei, Y. Yiven, "Mining Frequent Patterns without Candidate Generation," *ACM SIGMOD Record*, vol. 29(2), pp. 1–12, 2000. doi: 10.1145/342009.335372
20. S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," *Journal of Documentation*, vol. 60(5), pp. 503–520, 2004. doi: 10.1108/00220410410560582
21. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *International Conference on Learning Representations (ICLR), May 2-4, 2013, Scottsdale, Arizona, USA, 2013*. doi: https://doi.org/10.48550/arXiv.1301.3781
22. J. Pennington, R. Socher, and C.D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 25-29 October 2014, Doha, Qatar*, pp. 1532–1543, 2014. doi: 10.3115/v1/D14-1162
23. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. doi: 10.1162/tacl_a_00051
24. A. Grover, J. Lescovec, "node2vec: Scalable Feature Learning for Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 13-17, 2016, San Francisco, CA, USA*, pp. 855–864, 2016. doi: https://doi.org/10.1145/2939672.2939754

*Received 06.06.2024*

## INFORMATION ON THE ARTICLE

**Nadezhda I. Nedashkovskaya,** ORCID: 0000-0002-8277-3095, Educational and Research Institute for Applied System Analysis of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine, e-mail: nedashkovskaya.nadezhda@lll.kpi.ua

**Dmytro V. Androsov,** ORCID: 0009-0001-1330-1473, Educational and Research Institute for Applied System Analysis of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine, e-mail: androsovdmitry80@gmail.com

**ГІБРИДНИЙ МЕТОД ПОБУДОВИ СЕКВЕНЦІАЛЬНИХ РЕКОМЕНДАЦІЙНИХ СИСТЕМ, ЗАСНОВАНИЙ НА АВТОМАТИЧНОМУ СИНТЕЗІ ГРАФІВ ЗНАНЬ ТА МЕХАНІЗМІ УВАГИ** / Д.В. Андросов, Н.І. Недашківська

**Анотація.** Послідовні персоналізовані рекомендації, такі як прогнозування наступної найкращої пропозиції або моделювання еволюції попиту для прогнозування наповнення кошика покупок, залишаються ключовим завданням для бізнесу. Останнім часом, задля вирішення цих проблем застосовувалися ланцюги Маркова та моделі глибокого навчання, що прогнозували послідовності взаємодії користувачів із товарами, демонстрували високу ефективність. Проте ключовим недоліком таких моделей було неуніфіковане подання наборів даних для довгострокового та короткострокового прогнозуваннння вподобань. З появою архітектур глибокого навчання на графах та можливості їх застосування одночасно в задачах колаборативної фільтрації та прогнозування зв'язків між об'єктами, розвиток рекомендаціних систем отримав новий поштовх. Пропоновано новий метод розроблення гібридних рекомендаційних систем, який поєднує навчання подань графів з глибокими нейронними мережами для моделювання та прогнозування послідовностей, з метою розв'язання задачі видачі послідовних персоналізованих рекомендацій. Отримані результати оцінювання продуктивності моделі на основі набору даних відвідувань та купівель в інтернет-магазині доводять можливість ранжування та потенціал для впровадження бізнесами у сфері роздрібної торгівлі.

**Ключові слова:** рекомендаційна система, графова нейронна мережа, подання графів.