DOI: 10.17323/2587-814X.2025.2.41.53

A customer avatar model based on Kolmogorov–Arnold networks

Fedor V. Krasnov
E-mail: krasnov.fedor2@rwb.ru

Fedor I. Kurushin
E-mail: kurushin.fedor@rwb.ru

Research Center of LLC WILDBERRIES SK, Moscow, Russia

Abstract

The increasing pace of development of e-commerce continues to present new challenges in terms of personalizing product search and recommendations. Monolithic search and recommendation systems have become cumbersome and are unable to effectively address the need for a deeper understanding of users on electronic trading platforms (ETPs) despite having access to comprehensive information about their interests and purchase histories. Collaborative filtering mechanisms which are widely used suffer from a lack of diversity in offerings and a reduced capacity to surprise users. Additionally, the low frequency of recommendation updates and the replacement of "personalized" with "similar to others" concepts contribute to these issues. We have approached the resolution of these issues by developing a shopping assistant named "Ellochka" that is individual for each user of ETP. The digital avatar model of the user continually searches for relevant products based on their history of interaction with ETP. We were guided by the principle of independence – avatar models do not share information with each other. When a new user joins, they are assigned a unique avatar model that evolves independently. Each avatar has its own language to generate search queries. The

level of complexity of each avatar can vary depending on the intensity of its interaction with ETP. Continued interaction with the avatar allows for tracking of optimal purchase conditions, reminding users of expiration dates and the need for re-purchasing frequently purchased items. Isolating the avatar allows it to be retrained after each event, without significantly impacting the overall search and recommendation system. The use of neural network architecture-based and Kolmogorov–Arnold networks in the avatar-model has led to improvements in the main indicators of search and recommendation effectiveness, namely, novelty and diversity.

Keywords: large language models, product search, search query recommendations, search query transformation, user intent determination, text analysis, machine learning, e-commerce

Citation: Krasnov F.V., Kurushin F.I. (2025) A customer avatar model based on Kolmogorov–Arnold networks. *Business Informatics*, vol. 19, no. 2, pp. 41–53. DOI: 10.17323/2587-814X.2025.2.41.53

Introduction

Artificial intelligence is increasingly penetrating scientific methodology and everyday life. On the one hand, approaches to the classification of scientific knowledge are being revised; for example, a section "28.23: artificial intelligence" has appeared in the Code of State Categories Scientific and Technical Information. On the other hand, in an online store of women's clothing, the science-intensive processes are comparable to office work in geology [1].

This paper presents an avatar model of a personal assistant for making purchases on electronic trading platforms (ETP) called "Ellochka," in honor of the heroine of the work by Ilf and Petrov "12 Chairs". In this novel, Ellochka is a beautiful and spoiled young woman who lives for her own pleasure at her husband's expense and is mainly engaged in purchasing things.

Personal assistants based on artificial neural networks of deep learning have become a familiar attribute of customer service in the logistics, banking and e-commerce industries. However, the "personality" of such an assistant is limited only by the temporary desire to save resources during the maintenance session. The user's personal preferences and moods do not affect the operation of the system in any way. The personal assistant will not be able to continue the interrupted conversation from the same place and, most likely, will forget its beginning.

Personal assistants and search and recommendation models in e-commerce are based on large language models (LLM) based on the transformer architecture [2].

However, each large language model contains a dictionary file which it uses further in neural network structures in the form of ordinal numbers of tokens from the dictionary. This is because linear algebra works with models of numbers rather than string variables.

The size of the LLM dictionary largely determines the number of model parameters, because it is one of the main dimensions of the LLM along with the size of the model context window. *Table 1* shows the sizes of the most effective dictionaries according to the MTEB benchmark [3].

Table 1.

Industry research indicators

Model	Vocabulary size	Number of Russian words
alan-turing-institute/mt5-large-finetuned-mnli-xtreme-xnli	250 100	26 427
ai-forever/rugpt3large_based_on_gpt2	50 257	43 213
aeonium/Aeonium-v1-Base-4B	128 000	102 913

Search and recommendation systems are highly reliable information systems that ensure the continuity of the service process and minimal delays in interacting with buyers and sellers.

Speed in serving search queries is achieved through horizontal scaling of computing resources. A specific computing resource is not assigned within a single user service session. This leads to the need to synchronize user actions that change, for example, the balance, stock balances or the contents of the customer's shopping cart. These digital objects exist in the singular and are managed centrally.

The main contribution of this study is the formulation and testing of a new, decentralized approach to building search and recommendation systems in e-commerce, focusing on the avatar model of the buyer (*Fig. 1*).

The second most important contribution of this study is the expansion of the approach to building artificial deep learning networks for time series processing by a new class of Kolmogorov–Arnold networks using wavelet transformations.

1. Methodology

In order to avoid the well-known problems with novelty and diversity in product recommendations [4–7], in this study, we propose to consider the rec-

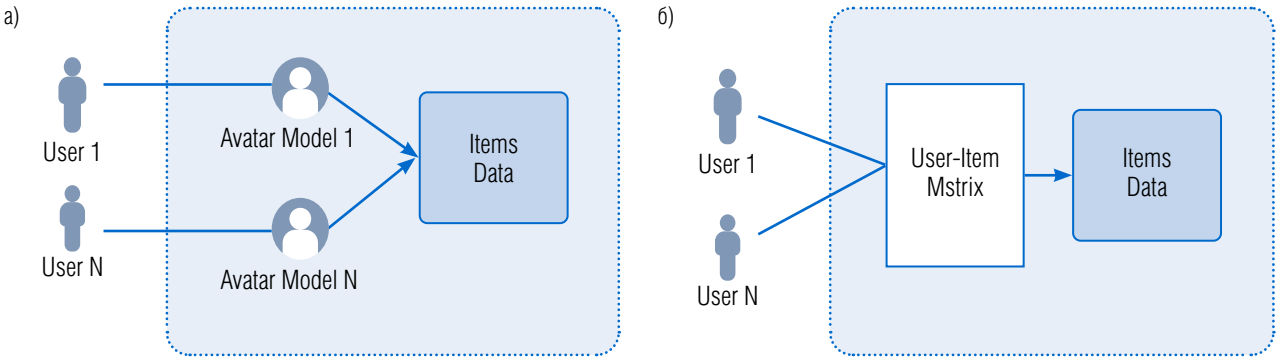

Fig. 1. The scheme of the recommendations:
(a) Decentralized (proposed in this study), (b) Centralized.

ommendations of search queries that lead to the right products. In everyday terms, you recommend how to search for a product, rather than trying to guess the desired product. A similar approach to recommendations using text is proposed in [8], in which the authors receive a significant improvement in metrics by rejecting product recommendations by their number, as in [9, 10].

Another well-known problem of generative models based on LLM is hallucination. In order to overcome the negative effects of hallucination, various methods are used to regularize the output of the model. For example, the reflection of the model on itself [11] or the use of adaptive spectral normalization [12]. The authors of this study have reduced the negative effect of hallucination due to the fact that the generated search query text is not intended for a human, but for a search engine, in other words, for another machine learning model. This made it possible to reduce the requirements for vocabulary and coherence of the generated search query text, since the presence of tokens is sufficient for a correct semantic search for products; in turn, the inconsistency of tokens by case or repetition affects the relevance of the search only slightly.

The avatar model is based on the paradigm of information search – the composition of candidate and ranking models [13]. The authors of this study developed this approach and, instead of a ranking model, created a model for selecting pairs of candidate queries with fixed five levels of relationships: "narrowing," "expanding," "paraphrasing," "different characteristics," "related products." This made it possible to create more customer-focused interaction scenarios, depending on the presence or absence of purchases in the interaction session with the ETP.

Modern tokenization models in LLM are based on compressed dictionaries from character n-grams [14, 15]. This approach solves the problem of missing a word in the dictionary and reduces memory consumption, but tokenization is ambiguous – the same sentence can be tokenized into different sets of tokens. This property is used in LLM training as bootstrapping for a more uniform distribution of the back propagation of the error. For the avatar model, there is no need to solve the problem of tokenizing new words, since the avatar model uses all tokens from the user's vocabulary. Moreover, the avatar model uses dictionary n-grams up to four words long to receive events from token structures. For example, from brand names consisting of several words, such as "Vans off the Wall." Obviously, the search query "Vans off the Wall" should not be used to search for products with the tokens "vans" and "wall."

When updating the avatar model, the new tokenization model is created, since the avatar model has 103 times fewer training parameters than, for example, the GPT2 (120 million) model.

2. Kolmogorov–Arnold networks

Kolmogorov–Arnold networks (KAN) show promising results in various generative models [16] and time series models [17]. We will show the advantages of KAN in comparison with the Multilayer Rumelhart Perceptron (MLP). Let's give an MLP with dimension n at the input and m at the output, containing fully connected layers of an artificial neural network from l to $l + 1$. Then the MLP equation in matrix form is given by formula (1):

$$x^{(l+1)} = W_{l+1,l} x^{(l)} + b^{(l+1)}, \quad (1)$$

where $W_{l+1,l}$ is the weight matrix connecting the layer and the layer ;

$x^{(l)}$ is the input vector;

$x^{(l+1)}$ is the output vector;

$b^{(l+1)}$ is the bias for $l + 1$.

Weight matrix $W_{l+1,l}$ can be represented as follows:

$$W_{l+1,l} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix}, \quad (2)$$

where $w_{i,j}$ is the weight between the i node in the $l+1$ layer and the j node in the l layer, $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$.

Then the bias vector $b^{(l+1)}$ can be presented as follows:

$$b^{(l+1)} = \begin{pmatrix} b_1^{(l+1)} \\ b_2^{(l+1)} \\ \vdots \\ b_m^{(l+1)} \end{pmatrix}. \quad (3)$$

Thus, in the expanded form, equation (1) looks like this:

$$\begin{pmatrix} x_1^{(l+1)} \\ x_2^{(l+1)} \\ \vdots \\ x_m^{(l+1)} \end{pmatrix} = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{pmatrix} \begin{pmatrix} x_1^{(l)} \\ x_2^{(l)} \\ \vdots \\ x_n^{(l)} \end{pmatrix} + \begin{pmatrix} b_1^{(l+1)} \\ b_2^{(l+1)} \\ \vdots \\ b_m^{(l+1)} \end{pmatrix}. \quad (4)$$

Now let's assume that we have L layers, each of which has a structure (4). Let's say $\sigma(\cdot)$ is an activation function. Then the compact formula for the entire MLP network $f(x)$, where x is the input vector, and $f(\cdot)$ is the MLP, is given as follows (5):

$$f(x) = x^{(L)}. \quad (5)$$

where

$$x^{(l+1)} = \sigma(W_{l+1,l} x^{(l)} + b^{(l+1)}), \quad (6)$$

for $l = 0, 1, 2, ..., L-1$, $x^{(l+1)}$ denotes the output vector.

This is equivalent to the following representation (7):

$$f(x) = \sigma \begin{pmatrix} W_L \sigma(W_{L-1} \cdots \sigma(W_2 \sigma(W_1 x + b_1) + \\ + b_2) \cdots + b_{L-1}) + b_L \end{pmatrix}. \quad (7)$$

Now let's look at how relationships between layers are created in KAN. Let $x^{(l)}$ be a vector with dimension n. Then the transposed vector can be represented as follows:

$$x^{(l)} \in R^n \Rightarrow (x^{(l)})^{\mathrm{T}} \in R^{1 \times n}.$$

Let's denote the matrix of transposed vectors $x^{(l)}$ with m rows and n columns as X_l:

$$X_l = \begin{pmatrix} (x^{(l)})^{\mathrm{T}} \\ (x^{(l)})^{\mathrm{T}} \\ \vdots \\ (x^{(l)})^{\mathrm{T}} \end{pmatrix} \in R^{m \times n}.$$

Each row of the matrix X_l – is a transposed vector $(x^{(l)})^{\mathrm{T}}$. Now let's define the operator A_o, that acts on the matrix $\Phi_{l+1,l}(X_l)$. This operator summarizes the elements of each row of the matrix and outputs the resulting vector r. Thereby definition formula of A_o:

$$A_o(\Phi_{l+1,l}(X_l)) = r,$$

where r vector, derived from the following expression (8):

$$r_i = \sum_j \left[\Phi_{l+1,l}\left(X_l\right) \right]_{ij} = \sum_{j=1}^{n} \phi_{i,j}\left(x_j^{(l)}\right), \text{ for } i = 1, 2, ..., m. \quad (8)$$

In the expression (8), $\left[\Phi_{l+1,l}\left(X_l\right) \right]_{ij}$ corresponds to the element in the i row and j column of the matrix $\Phi_{l+1,l}(X_l)$. Thus, the operator A_o can be written as follows:

$$A_o\left(\Phi_{l+1,l}\left(X_l\right)\right) = \sum_j \left[\Phi_{l+1,l}\left(X_l\right) \right]_{ij}.$$

By definition A_o performs an action on a matrix $\Phi_{l+1,l}(X_l)$, summarizes the elements in each row, and outputs the resulting vector r. Indeed, Φ_{l+1} receives a vector at the input $x^{(l)}$ and outputs data at the output, where each element of $x^{(l)}$ is the sum of one element (9):

$$X_{l+1,l} = \Phi_{l+1,l}(X_l), \quad (9)$$

where:

$$\Phi_{l+1,l}\left(X_l\right) = \begin{pmatrix} \phi_{1,1}(x_1^{(l)}) & \phi_{1,2}(x_2^{(l)}) & \cdots & \phi_{1,n}(x_n^{(l)}) \\ \phi_{2,1}(x_1^{(l)}) & \phi_{2,2}(x_2^{(l)}) & \cdots & \phi_{2,n}(x_n^{(l)}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{m,1}(x_1^{(l)}) & \phi_{m,2}(x_2^{(l)}) & \cdots & \phi_{m,n}(x_n^{(l)}) \end{pmatrix}. \quad (10)$$

In expression (10) $\Phi_{l+1,l}$ is the activation function between the layers l and $l + 1$. Each element $\phi_{i,j}(\cdot)$ denotes an activation function that connects the j neuron in the l layer with the neuron in the $l + 1$ layer. Instead of multiplication, equation (10) calculates a function with trainable parameters. Therefore, if we consider X_o as input data that contains only the input vector in the form of strings, then for the entire KAN, the output data after the L layers will be as follows (11):

$$f_{KAN}\left(X_0\right) = x^{(L)} = A_o\left(\Phi_{L,L-1}\left(X_{L-1}\right)\right) =$$

$$= A_o\left(\Phi_{L,L-1}\left(\begin{pmatrix}\left(A_o\left(\Phi_{L-1,L-2}\left(X_{L-2}\right)\right)\right)^{\mathrm{T}}\\ \left(A_o\left(\Phi_{L-1,L-2}\left(X_{L-2}\right)\right)\right)^{\mathrm{T}}\\ \vdots \\ \left(A_o\left(\Phi_{L-1,L-2}\left(X_{L-2}\right)\right)\right)^{\mathrm{T}}\end{pmatrix}\right)\right) = \quad . \quad (11)$$

$$= A_o\left(\Phi_{L,L-1}\left(\begin{pmatrix}\left(A_o\left(\Phi_{L-1,L-2}\cdots\left(A_o\left(\Phi_{1,0}\left(X_0\right)\right)\right)\right)\right)^{\mathrm{T}}\\ \left(A_o\left(\Phi_{L-1,L-2}\cdots\left(A_o\left(\Phi_{1,0}\left(X_0\right)\right)\right)\right)\right)^{\mathrm{T}}\\ \vdots \\ \left(A_o\left(\Phi_{L-1,L-2}\cdots\left(A_o\left(\Phi_{1,0}\left(X_0\right)\right)\right)\right)\right)^{\mathrm{T}}\end{pmatrix}\right)\right)$$

Thus, traditional MLPs use fixed nonlinear activation functions at each node, linear weights and biases to transform input data across layers. The output data on each layer is calculated using a linear transformation followed by a fixed activation function. During the backpropagation of the error, the gradients of the loss function relative to the weights and biases are calculated to update the model parameters. In contrast, KAN replaces linear weights with one-dimensional functions that can be trained by placing them on edges rather than on nodes. Wavelet transformations are most often used as one-dimensional functions: MHAT wavelet ("Mexican hat"), Shannon wavelet, Morlet wavelet, Gauss wavelet (*Table 2*).

One-dimensional functions from previous levels are summed up in the nodes. Each function can be adapted, which allows KAN to train both activation and transformation of input data. This change leads to increased accuracy and interpretability, since KAN can better approximate functions using fewer parameters. During backpropagation of an error in KAN, gradients are calculated relative to one-dimensional functions, updating them to minimize the loss function. This leads to more efficient learning for complex and multidimensional functions.

Table 2.

Wavelets used as one-dimensional functions

Name	Formula
MHAT-wavelet ("Mexican hat")	$\psi(t) = \frac{-d^2}{dt^2} e^{-t^2/2} = \left(1 - t^2\right) e^{-t^2/2}$
Shannon wavelet	$\psi_k^n(t) := 2^{n/2} \psi^{(Sha)}\left(2^n t - k\right)$, where $\psi^{(Sha)}(t) := \frac{\sin\left(\pi t\right)}{\pi t}$
Morlet wavelet	$\psi_\sigma(t) = c_\sigma \pi^{-1/4} e^{-t^2/2}\left(e^{i\sigma t} - k_\sigma\right)$, where $k_\sigma = e^{-\sigma^2/2}$, $c_\sigma = \left(1 + e^{-\sigma^2} - 2e^{-3\sigma^2/4}\right)^{-\frac{1}{2}}$
Gauss wavelet	$\psi(t) = \frac{-d}{dt} e^{-t^2/2}$

3. A model for reducing the diversity of search queries

In the field of e-commerce, there is the problem of the "long tail" of search queries, which makes it difficult to create effective models. This is due to a wide variety of queries, typos, synonyms and slang.

Modern approaches to query reformulation use neural network methods. By highlighting text factors, vector representations of search queries are constructed, and then the "nearest neighbors" are searched in vector space using Retrieval Augmented Generation (RAG) language models [18] and BERT transformer models [19].

Product search allows you to use user behavioral data from making purchases. Thus, a collaborative paradigm is used. The user-product relationship determines the sequences that led to the purchase and uses them as options for the current user [20]. In the present study, the problem of query reformulation (QR) is considered from the perspective of the "search query – product" relationship for the application of a surrogate function [22] in the avatar model.

In addition to the reformulation of requests, the task is also to determine the type of connection "request – request" and "request – product". As an example, demonstrating the various types of relationships identified by the authors, we present *Table 3*.

For a more visual representation, let's denote the set of all search queries as Q. The approximate number of elements in this set is $Q_v \approx 10^{10}$, which is comparable to the stream of events analyzed in the search for the Higgs boson [21].

It is known from the study [23] that 98% of all purchases are made using a limited set of queries, which we will denote as Q_{HPQ}. Without limiting generality, we can assume that Q_{HPQ} does not depend on time. The remaining set of queries is denoted as Q_{LPQ}, where HPQ and LPQ are common abbreviations for high—performing and low-performing queries, respectively.

Based on the definition of the set Q, we can write the following equation:

$$Q = Q_{HPQ} \cup Q_{LPQ}, \tag{12}$$

$$\varnothing = Q_{HPQ} \cap Q_{LPQ}. \tag{13}$$

Table 3.

Examples of relation between the pairs of queries

Source Query	Suggest	Scenario
Iphone 16 pro max	Apple iPhone 16 Pro Max	Paraphrased
Women's red Dress	Women's Dress	Expanding
Dress	Women's red Dress	Narrowing
Women red dress	Women's blue dress	Other characteristics
Women's evening dress	High-heeled shoes	Substituting
High-heeled shoes	Running shoes	Not relevant

It follows from the nature of the Q distribution that $|Q_{LPQ}| \gg |Q_{HPQ}|$. The task of reformulating queries can then be considered as a search for a function F such that $F(Q_{LPQ}) \Rightarrow Q_{HPQ}$. Consider the requests $q_{HPQ}^{p_i} \in Q_{HPQ}$ and $q_{LPQ}^{p_i} \in Q_{LPQ}$ resulting purchases of a product $p_i \in P$ from the entire product catalog P. Then we can write that there is a surrogate function F that results $q_{LPQ}^{p_i} q_{HPQ}^{p_i}$ for the product p_i.

$$F\left(Q_{LPQ}^{p_i}\right) \Rightarrow Q_{HPQ}^{p_i}, \text{ for } \forall p_i \in P. \quad (14)$$

Expression (14) allows us to formulate the conditions for obtaining a data set for using numerical methods of obtaining F, for example, using artificial neural networks of deep learning. To do this, you need to collect pairs $\left\{q_{LPQ}^{p_i}, q_{HPQ}^{p_i}\right\}$ from user search logs.

The Negative Batch Sampling approach was used to generate negative examples [10], with the number of negative examples varying from 5 to 12. Using a large number of negative examples improved the model's performance slightly, but required significantly more resources to train the model.

In *Fig. 2* shows a diagram of the avatar model in the form of a sequence of tokenization models, query reformulation (QR) models, and KAN.

4. Research questions

This research is applied in nature. The described methodology serves as a source for research questions and verification using numerical methods. When creating the methodology, the authors formulated the following research questions:

RQ-1: What is the size distribution of the dictionary used by users when forming search queries?

RQ-2: How does the use of KAN quantitatively improve the avatar model compared to MLP?

5. Experiment

To test our methodology, we collected a sequence of purchase search queries for random customers over the year. The result is a dataset D with a unique key "user" and a time series ["request1," ..., "purchase1," "end of session," "request N"]. Each purchase is presented as a text consisting of the name, brand and product characteristics. Typos have been corrected in search queries, synonyms have been added, and low-performance queries have been mapped to high-performance queries. This way, a time-ordered text consisting of queries and descriptions of purchased goods is received for each user in D. Based on this text, autoregressive training of the avatar model of each user is performed.

Using the obtained avatar model, a predictive model was built for each user: based on the entered search queries, the next search query from the search query index is predicted. Pessimization is performed on words from purchased goods, taking into account their remoteness in terms of user interaction time. Multiple search queries are predicted for the user.

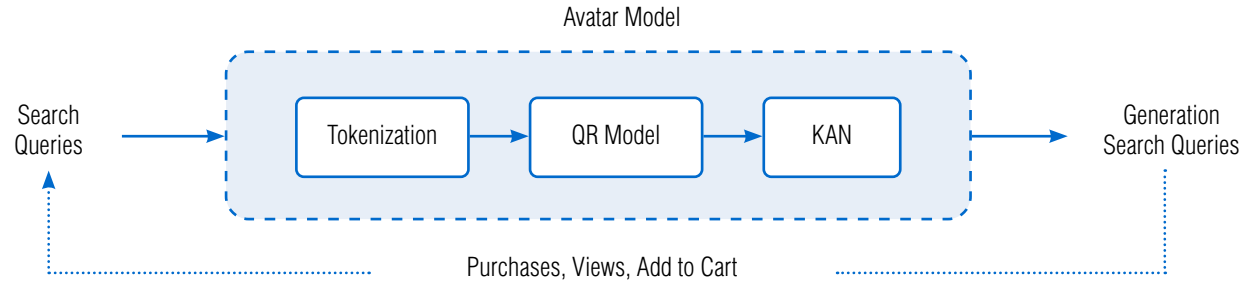

Fig. 2. Avatar model scheme.

Here is a scenario for using the results of this study: A user visits an ETP, does not enter a search query, sees search results for products that are close to him in vocabulary and do not contain recent purchases in the first place. The search output reflects the current status of the product catalog, inventory, region, popularity and other conditions that enhance relevance. The search results are highly diverse and new, since they contain products from several predicted queries for the user. The issue does not contain the purchased goods in the first positions.

For an experimental answer to the RQ-1 research question, a distribution of the number of unique words for each user was constructed from the dataset D (*Fig. 3*).

From the distribution in *Fig. 3*, we obtain that the maximum dictionary size does not exceed 1500 words. Compared to the size of the dictionary in the LLM shown in *Table 1*, the size of the dictionary of an individual user is an order of magnitude smaller. This fact provides a basis for a significant acceleration of the avatar model compared to the LLM.

To find an answer to the research question RQ-2, a series of experiments were conducted to train an avatar model on a data set D. The following options were considered as hyperparameters of the avatar model:

Tokenisation model:

Unigram, byte pair encoding (BPE), **Word**

Vocabulary size: 100, 1000, **"Unlimited"**

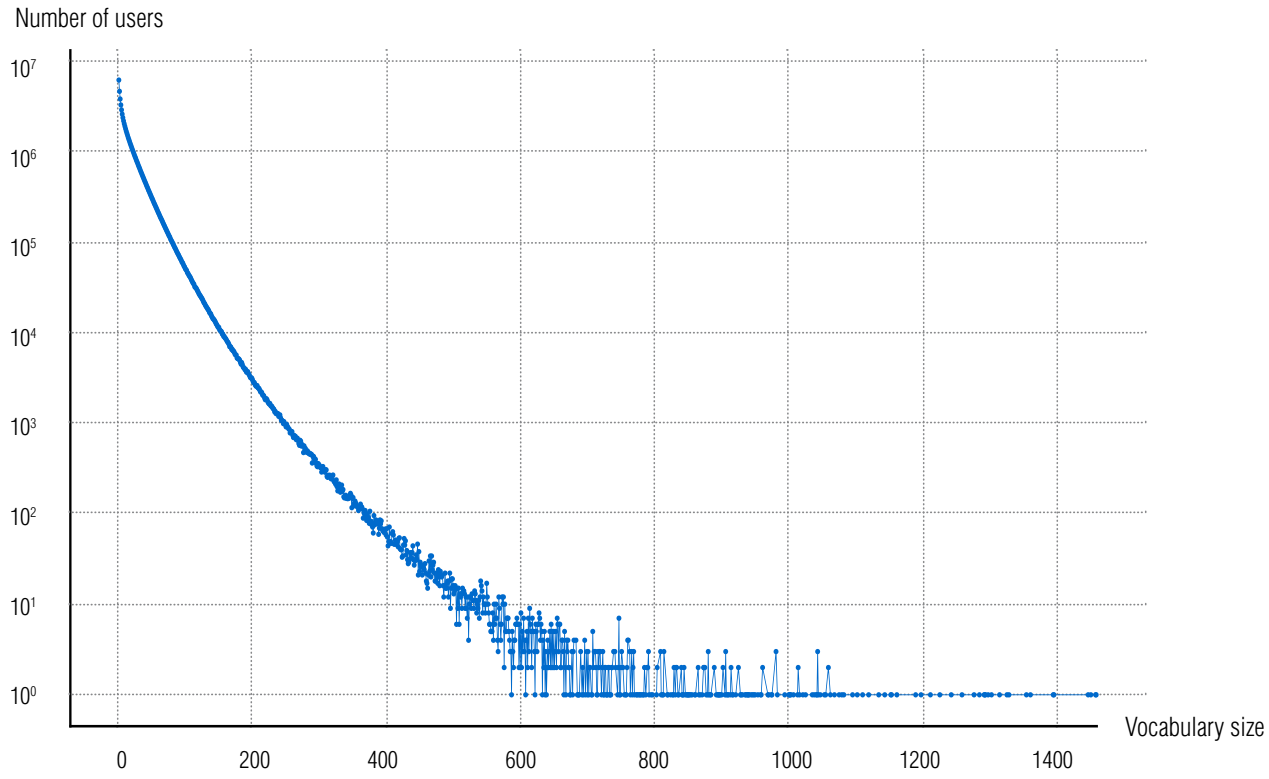

Fig. 3. Distribution of the number of unique words for each user.

Number of examples for subword tokenization: 3, 5, 7, 9, **without sampling**

QR Model:

The dimension of the vector space of the token representation: 32, **64**, 128, 256

Recurrent Neural Network (RNN) parameters:

Number of layers: **2**, 3, 4, 5, 6

Bidirectional: yes, **no**

Type: LSTM, **GRU**

In-batch negative sampling: 3, **5**, 7, 12

Dual Margin Cosine Embedding Loss: positive margin **0.9**, negative margin **0.2**

Shared vector embeddings: no, **yes**

KAN model:

Wavelet type: **"Mexican hat",** Shannon wavelet, Morlaix wavelet, Gauss wavelet.

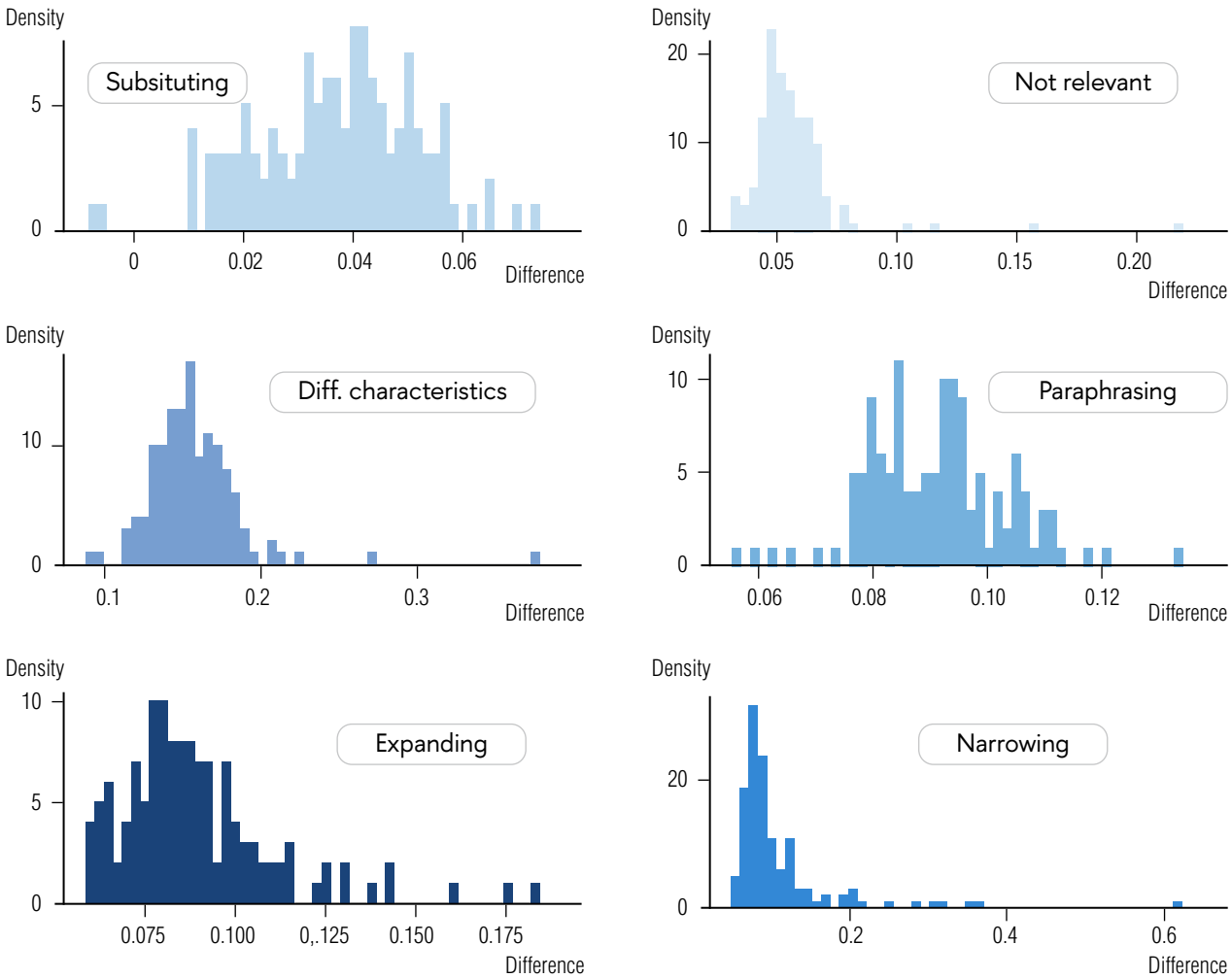

Fig. 4. Distribution of differences in the Accuracy metric between SKA and MPR.

A total of 120 experimental sessions were conducted, each of which lasted from 12 to 18 hours. The study identified the parameters for which the avatar model demonstrated the best results in terms of convergence rate and minimal learning error.

All other things being equal, the distribution shown in *Fig. 4* shows positive changes in accuracy when using KAN in the avatar model in comparison with MLP.

Conclusion

This study suggests a new approach to creating a recommendation system for users in the field of e-commrce, named "Ellochka" in honor of the heroine of the novel "12 Chairs" with a small vocabulary, who successfully coped with any communication tasks.

The authors have developed and tested a methodology based on the following principles:

1. Abandoning the use of a monolithic, unified recommendation system for all users of the electronic trading platform in favor of creating separate recommendation models for each user.

2. Building language models for recommending search query texts based on small-size token dictionaries instead of huge dictionaries with character tokenization.

3. Application of the mathematical apparatus of Kolmogorov–Arnold networks to improve the convergence rate of models during training.

The methodology proposed in the article has been successfully tested on the data of a working electronic trading platform and has allowed us to improve the autonomous indicators of the recommendation system of tips. ■

References

1. Butorin A.V., Murtazin D.G., Krasnov F.V. (2020) *Method and system for predicting effective thicknesses in the interwell space when constructing a geological model based on the method of clustering spectral curves*. Patent for invention RU 2718135 C1, 03/30/2020. Application No. 2019128334 dated 09/09/2019.

2. Krasnov F. (2023) Query understanding via Language Models based on transformers for e-commerce. *International Journal of Open Information Technologies*, vol. 11, no. 9, pp. 33–40 (in Russian).

3. Muennighoff N., Tazi N., Magne L., Reimers N. (2022) MTEB: Massive text embedding benchmark. *arXiv:2210.07316*. https://doi.org/10.48550/arXiv.2210.07316

4. Li P., Tuzhilin A. (2023) When variety seeking meets unexpectedness: Incorporating variety-seeking behaviors into design of unexpected recommender systems. *Information Systems Research*, vol. 35, no. 3. https://doi.org/10.1287/isre.2021.0053

5. Wang Y., Banerjee C., Chucri S., et al. (2024) Beyond item dissimilarities: Diversifying by intent in recommender systems. *arXiv:2405.12327*. https://doi.org/10.48550/arXiv.2405.12327

6. Castells P., Hurley N., Vargas S. (2022) Novelty and diversity in recommender systems. *Recommender Systems Handbook* (eds. F. Ricci, L. Rokach, B. Shapira). Springer, New York, NY, pp. 603–646. https://doi.org/10.1007/978-1-0716-2197-4_16

7. Ding Q., Liu Y., Miao C., et al. (2021) A hybrid bandit framework for diversified recommendation. Proceedings of the *AAAI conference on artificial intelligence*, vol. 35, no. 5, pp. 4036–4044. https://doi.org/10.1609/aaai.v35i5.16524

8. Li J., Wang M., Li J., et al. (2023) Text is all you need: Learning language representations for sequential recommendation. Proceedings of the *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1258–1267. https:// doi.org/10.1145/3580305.3599519

9. Sun F., Liu J., Wu J., et al. (2019) BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. Proceedings of the *28th ACM international conference on information and knowledge management*, pp. 1441–1450. https://doi.org/10.1145/3357384.3357895

10. Klenitskiy A., Vasilev A. (2023) Turning dross into gold loss: is BERT4Rec really better than SASRec? Proceedings of the *17th ACM Conference on Recommender Systems*, pp. 1120–1125.

11. Ji Z., Yu T., Xu Y., et al. (2023) Towards mitigating LLM hallucination via self reflection. *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1827–1843.

12. Egorov E.A., Rogachev A.I. (2023) Adaptive spectral normalization for generative models. *Doklady Mathematics*, vol. 108(suppl. 2), pp. S205–S214. https://doi.org/10.1134/S1064562423701089

13. Chang W.C., Jiang D., Yu H.F., et al. (2021) Extreme multi-label learning for semantic matching in product search. Proceedings of the *27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2643–2651.

14. Sennrich R., Haddow B., Birch A. (2016) Neural machine translation of rare words with subword units. Proceedings of the *54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725. https://doi.org/10.18653/v1/P16-1162

15. Kudo T., Richardson J. (2018) SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. Proceedings of the *2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71.

16. Liu Z., Wang Y., Vaidya S., et al. (2024) KAN: Kolmogorov-Arnold networks. *arXiv:2404.19756*. https://doi.org/10.48550/arXiv.2404.19756

17. Vaca-Rubio C. J., Blanco L., Pereira R., Caus M. (2024) Kolmogorov-Arnold networks (KANs) for time series analysis. *arXiv:2405.08790*. https://doi.org/10.48550/arXiv.2405.08790

18. Ma X., Gong Y., He P., et al. (2023) Query rewriting for retrieval-augmented Large Language Models. *arXiv:2305.14283*. https://doi.org/10.48550/arXiv.2305.14283

19. Chen Z., Fan X., Ling Y. (2020) Pre-training for query rewriting in a spoken language understanding system. *I2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020), Barcelona, Spain*, pp. 7969–7973. https://doi.org/10.1109/ICASSP40776.2020.9053531

20. Bhandari M., Wang M., Poliannikov O., Shimizu K. (2023) RecQR: Using Recommendation Systems for Query Reformulation to correct unseen errors in spoken dialog systems. *17th ACM Conference on Recommender Systems (RecSys'23), Singapore*, pp. 1019–1022.

21. Kim T.H., Neubauer M., Sfiligoi I., et al. (2004) The CDF central analysis farm. *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 892–896. https://doi.org/10.1109/TNS.2004.829574

22. Kuleshov A.P. (2008) Cognitive technologies in adaptive models of complex objects. *Informatsionnye Tekhnologii i Vychslitel'nye Sistemy*, vol. 1, pp. 18–29.

23. Luo C., Lakshman V., Shrivastavaet A., et al. (2022) ROSE: Robust caches for Amazon product search. *WWW '22: Companion Proceedings of the Web Conference 2022*, pp. 89–93. https://doi.org/10.1145/3487553.3524213

About the authors

Fedor V. Krasnov

Candidate of Sciences (Technology);

Senior Data Scientist, Research Center of WB SK LLC based on the Skolkovo Innovation Center, 5, Nobel St., Mozhaisk District, Western Administrative District, Moscow, Russia;

E-mail: krasnov.fedor2@rwb.ru

ORCID: 0000-0002-9881-7371

Fedor I. Kurushin

Doctoral Student;

Data Scientist, Research Center of WB SK LLC based on the Skolkovo Innovation Center, 5, Nobel St., Mozhaisk District, Western Administrative District, Moscow, Russia;

E-mail: kurushin.fedor@rwb.ru

ORCID: 0009-0007-5126-4507