Bayesian click model and methods of estimating its parameters

Andriy Sverstiuk[a,b], Taras Dubynyak[b], Oleksandra Manziy[c], Andriy Senyk[c], Pavlo Ohloblin[c]

[a]*I. Horbachevsky Ternopil National Medical University, Maidan Voli, 1, Ternopil, 46002, Ukraine*
[b]*Ternopil National Ivan Puluj Technical University, Rus'ka str. 56, Ternopil, 46001, Ukraine*
[c]*National University "Lviv Polytechnic", S.Bandera str, 12, Lviv, 79013, Ukraine*

Abstract

In this article, mathematical support for the analysis of user click activity data has been developed. Based on the Bayesian click model and existing methods of estimating its parameters, a software product was created using the Python programming language, which predicts the relevance of web documents based on click logs. Studies have been conducted and conditions have been established under which the smallest error of these predictions is achieved.

Keywords

Mathematical methods, click model, data analysis, prediction, prediction error, algorithmic support, click logs, graph, probabilistic parameters, Bayesian network, machine learning.

1. Introduction

One of the metrics of user feedback in search and marketing systems is Click-Through Rate (CTR) – click rating or click ratio. The value of this indicator gives information about the interest of users in certain search results. Therefore, the tasks of calculating, evaluating and predicting this metric are important and relevant [1-3]. For mathematical modeling and analysis of clicks, so-called click models have been developed, which are described by a set of probability relationships [4]. Research on this topic is presented in the works of scientists from Canada (Zhe Gao and Qigang Gao from the Faculty of Computer Science, Dalhousie University, Canada), Korea (Kyungwon Kim, Eun Kwon and Jaram Park from AI Center of Samsung Research, Samsung Electronics Company, Ltd., Republic of Korea) and other outstanding technical scientists [5-10].

Here are examples of the most common software products, services and online tools that are used to analyze data about user click activity:

1. Google Keyword Planner.
2. Microsoft Keyword Planner.
3. Facebook Campaign Planner.
4. LinkedIn Campaign Manager.
5. SellerApp.
6. Reddit Ads Dashboard.
7. Pinterest Ads Manager.

The development of new mathematical and software algorithms for analysis and predicting user click activity is an urgent task. The paper proposes the use of the Bayesian click model and the implementation on its basis of effective mathematical methods of analysis, evaluation and prediction of user actions for improving the quality and efficiency of search services, as well as analyzing user interaction with ads to select the most relevant of them.

ITTAP'2023: 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, November 22–24, 2023, Ternopil, Ukraine
EMAIL: sverstyuk@tdmu.edu.ua (Andriy Sverstiuk); d_taras@ukr.net (Taras Dubyniak), oleksandra.s.manzii@lpnu.ua (Oleksandra Manziy), andrij.p.senyk@lpnu.ua (Andrij Senyk), pavlo.ohloblin.mpmkm.2022@lpnu.ua (Pavlo Ohloblin).
ORCID: 0000-0001-8644-0776 (Andriy Sverstiuk); 0000-0003-1529-6951 (Taras Dubyniak), ORCID 0000-0002-6480-2307 (Oleksandra Manziy); ORCID 0000-0002-1614-512X (Andrij Senyk), ORCID 0009-0002-4515-1390 (Pavlo Ohloblin).

2. Mathematical grounding

Since CTR is characterized by a shift in the positions of the search results (the lower the result in the list, the lower its actual CTR), the task of its unbiased prediction arises. That is, the calculation of its value, which would not depend on the position of the result in the output. In addition, click models are used to calculate unbiased CTR values. Click models provide an opportunity to estimate the CTR of some web document in terms of probabilities. CTR is numerically equal to the ratio of the number of clicks on the banner or link to the total number of their impressions:

$$CTR = \frac{clicks}{impressions} \cdot 100\%$$

The main examples of such models are the following:

Random Click Model

This is the simplest model, which is described by the following equation:

$$P(C_u = 1) = \rho \quad (1)$$

where C_u is a random event – choose any URL u with equal probability ρ.

Position-based Model (PBM) [9]

More complex click models are based on the following hypothesis:

$$A_u = 1, E_u = 1 \Leftrightarrow C_u = 1 \quad (2)$$

where A_u - a random event - to become interested in a document, E_u - a random event - to get acquainted with its previous description (for example, a snippet in the search network). The ratio (2) means that the user clicks on the document u if and only if he has familiarized himself with this document and is interested in it. Random variables A_u and E_u are independent.

PBM is based on the assumption that the probability of getting acquainted with a snippet depends on the rank - the position of the document in the issue. The higher rank of the document, the lower its position and the corresponding probability of familiarization with it. The positional model describes this with the following ratios:

$$P(C_u = 1) = P(E_u = 1) \cdot P(A_u = 1), \quad (3)$$

$$P(A_u = 1) = a_u, \quad (4)$$

$$P(E_u = 1) = \gamma_u, \quad (5)$$

Cascade Model (CM)

The Cascade click model is based on the assumption that the user views the search results strictly from top to bottom and makes a click decision for each viewed document [11, 12]. After selecting the desired URL, the user will no longer view the documents below, regardless of their position. CM is described by the following ratios:

$$C_u = 1 \Leftrightarrow E_u = 1 \wedge A_u = 1, \quad (6)$$

$$P(A_u = 1) = a_u, \quad (7)$$

$$P(E_u = 1) = 1, \quad (8)$$

$$P(E_u = 1 | E_{u-1} = 0) = 0, \quad (9)$$

$$P(E_u = 1 | C_{u-1} = 1) = 0, \quad (10)$$

$$P(E_u = 1 | E_{u-1} = 1, C_{u-1} = 0) = 1. \quad (11)$$

Dynamic Bayesian Network (DBN) [12]

In [14], the authors proposed a Dynamic Bayesian Network - a click model, which is an extension of the cascade model (CM). The DBN for a fixed query is described by the following relations:

$$C_i = 1 \Leftrightarrow E_i = 1 \wedge A_i = 1, \quad (12)$$

$$P(A_i = 1) = a_u, \quad (13)$$

$$P(S_i = 1 | C_i = 1) = s_u, \quad (14)$$

$$C_i = 0 \Rightarrow S_i = 0, \quad (15)$$

$$S_i = 0 \Rightarrow E_{i+1} = 0, \quad (16)$$

$$P(E_{i+1} = 1 | E_i = 1, S_i = 0) = \gamma, \quad (17)$$

$$E_i = 1 \Rightarrow E_{i+1} = 0, \quad (18)$$

where *i* stands for the rank (position) of the document in the search results. C_i is a binary observable variable that shows whether a click occurred on the document at position *i*. The hidden variable E_i describes the fact of the user's familiarization with the snippet, A_i shows whether the user was interested in the document at position *i*. S_i shows whether the user was satisfied with the search result after visiting the web page at position *i*.

DBN is based on the following assumption: a click occurs if and only if the user has viewed a URL and is interested in it (12). The probability that the document will interest the user depends only on the document itself (13). Similar to the cascading model, the user browses URLs linearly from top to bottom until they decide to stop. Once a user clicks and visits a URL, there is a certain probability that they will be satisfied with the result of their search (14). On the other hand, if he does not browse the web page, he will not be satisfied with the search result (15). If the user is satisfied with the URL they visited, they stop searching (16). If he is not satisfied with the result, there is a probability $(1 - \gamma)$ that the user will stop searching (17) and a probability γ that the user will check the next URL. In other words, γ is a metric of user "persistence". If the user has not read the document snippet at position *i*, he will not explore the documents at lower positions (18). In addition, a_u and s_u are distributed according to the beta distribution.

The essence of learning click models is to estimate their parameters based on a set of data - the so-called click log. This data set contains information primarily about user searches, search results, and clicks on each of the results in the output. After evaluating the parameters of the model, it is possible to draw conclusions about the click behavior of users.

For DBN, the evaluated parameters are a_u and s_u, and these two parameters describe the relevance of document *u*. The parameter a_u describes hypothetical relevance as it measures the likelihood of a click based on a URL. The parameter s_u is equal to the probability that the user will be satisfied after going to this link; therefore s_u should be understood as a "ratio" between actual and hypothetical relevance.

For a dynamic Bayesian network, the parameter a_u is equivalent to the CTR that document *u* would have in the first position of the search results. Therefore, in the future, the main attention will be paid to the calculation of the parameter a_u, since it is a CTR forecast for the document *u* in the search results. At the same time, this CTR prediction is unbiased because a_u does not depend on the rank (position) of document *u* in the resulting list.

The EM algorithm and the Forward-Backwards algorithm [13] for DBN training, that is, for parameter estimation a_u and based on click logs s_u. They are used when implicit variables are present in the model. It is important to clarify that in the maximization step of the EM algorithm, the updated values of a_u and calculated s_u, and the maximum posterior method is used, which is a generalization of the maximum likelihood method. The easiest way to calculate the updated parameters is to calculate in closed form (analytically) using the theory of conjugate distributions. According to Bayes' theorem:

$$p(\theta|x) = \frac{p(x|\theta)\,p(\theta)}{\int_{range\theta} p(x|\theta)\,p(\theta)\,d\theta} \quad (19)$$

where for some parameter θ $p(\theta|x)$ - posterior distribution, $p(x|\theta)$ - likelihood function, $p(\theta)$ - a priori distribution. If the likelihood function and the prior distribution are conjugate, then $p(\theta|x)$ belongs to the same family of distributions as $p(\theta)$. Since in the Bayesian click model the likelihood function has a Bernoulli distribution and the prior probability has a beta distribution, the posterior probability will also have a beta distribution, but with different parameters (or

hyperparameters). Values of updated hyperparameters α and β of the posterior beta distribution are calculated as follows:

$$\alpha_{annocme} = \alpha_{anp} + \sum_{i=1}^{n} x_i \ , \ \beta_{annocme} = \beta_{anp} + \sum_{i=1}^{n} (1 - x_i) \, , \tag{20}$$

where x_i , $(i = \overline{1,n})$ are some independent observations.

The only input parameter that is given before starting DBN training is γ, the probability that the user will continue to view the output results at the positions $(i+1) = 1,...n$ provided that $S_i = 0$. [10] shows the graph of the root mean square error of predicting the CTR of the document at position 1 as a function of γ. As shown in Figure 3 [12], the best CTR prediction at position 1 was obtained for γ = 0.9. At the same time, the click model for γ = 1 gives only a slightly worse forecast, but at the same time the estimation of DBN parameters becomes much simpler. After all, in the case when γ = 1, the user continues to explore the search results until he is satisfied. This means that the last click in the list of all clicks gave the user a satisfactory result, and he stops reading snippets of documents that are in lower positions:

$$E_1 = ... = E_i = 1 \wedge E_{i+1} = ... = E_n = 0 \, , \tag{21}$$

where *i* is the position of the last clicked document. As can be seen from (21), for γ = 1 there is no variable uncertainty E_i , and therefore there is no need to apply EM and Forward-Backwards algorithms. Parameters a_u and s_u are estimated using simple calculations [12].

Therefore, the estimation of the parameter a_u , the algorithm of which is given above, is the essence of using the Bayesian click model (DBN) for the unbiased prediction of the CTR of the document *u*.

This work describes the software created by the authors, which implements the Bayesian click model for CTR predictions. Research has also been conducted and the conditions under which the smallest error of these predictions is achieved have been established.

3. Algorithmic support

A detailed analysis of the existing development tools was carried out and a decision was made to build a Bayesian click model using the Python programming language as the main tool, in particular its PyAgrum, NumPy, Pandas, Matplotlib libraries included in the Ananconda distribution. The Tkinter library was used to create the graphical interface. In addition, the software product Bayes Server is used as a tool for modeling conventional and dynamic Bayesian networks, causal models and influence diagrams. One of the most important capabilities provided by this software product is the calculation of the values of hidden variables. The main inference algorithm in Bayes Server is Relevance Tree, which is used by default. When calculating hidden variables, a choice is made: either to calculate the values of specific variables, or to calculate all of them at once. In addition, it is possible to calculate the value of the likelihood function. Another important feature of Bayes Server is parameter learning, for which it is necessary to additionally connect sets of training data. Bayes Server provides extensive options for editing models, for querying model parameters, including multiple queries based on large datasets.

Description of model inputs

Click logs are used to train click models. These are data sets that contain information about search sessions. A search session is a session during which a user makes a search request in the system and receives a result. After receiving the search results, the user goes through one or more links until he either finds the document he needs, or ends the search session due to the lack of relevant documents. Thus, each search session is characterized by three necessary attributes:

- search query;
- list of results;

- vector of clicks that takes values from {0; 1} and corresponds to the fact of clicking on a specific document.

Logs published in open access on the Kaggle resource as part of the international competition The Personalized Web Search Challenge were used to build and train the Bayesian click model. The data were formatted as illustrated in the Table 1:

Table 1. Format of click logs

query	url0	url1	...	url9	click 0	click 1	...	click 9
q_1	d_{10}	d_{11}	...	d_{19}	c_{10}	c_{11}	...	c_{19}
q_1	d_{11}	d_{10}	...	d_{19}	c_{11}	c_{10}	...	c_{19}
...	...	...	...	...	...	...	...	...
q_n	d_{n0}	d_{n1}	...	d_{n9}	c_{n0}	c_{n1}	...	c_{n9}

Here q_i $(i=\overline{1,n})$ – some search query; d_{ij} $(j=\overline{1,10})$ - a document (URL) associated with a search query q_i, c_{ij} $(j=\overline{1,10})$ – a Boolean variable associated with a document d_{ij}. If c_{ij} equal to 0, then the document was not clicked, if c_{ij} equal to 1 - it was clicked.

The original click logs are archived and have the *.gz format. To convert them into the format presented in the table. 1, a separate Python script file was created that implements the conversion of the archive to *.csv or *.sql. For this, the generator mechanism is used, which allows reading large and super-large arrays of data. From all the available information in the archive, only data of the following types are selected:

- session id;
- search request id;
- an identifier consisting of two numbers, the first is an anonymized address of some document, the second is its domain. This pair of numbers is read as one and interpreted as the URL of the document.

In addition, the log file lists (fig. 2) the URLs that were clicked on during this search session. For URLs from this list, the value 1 in the click vector is matched.

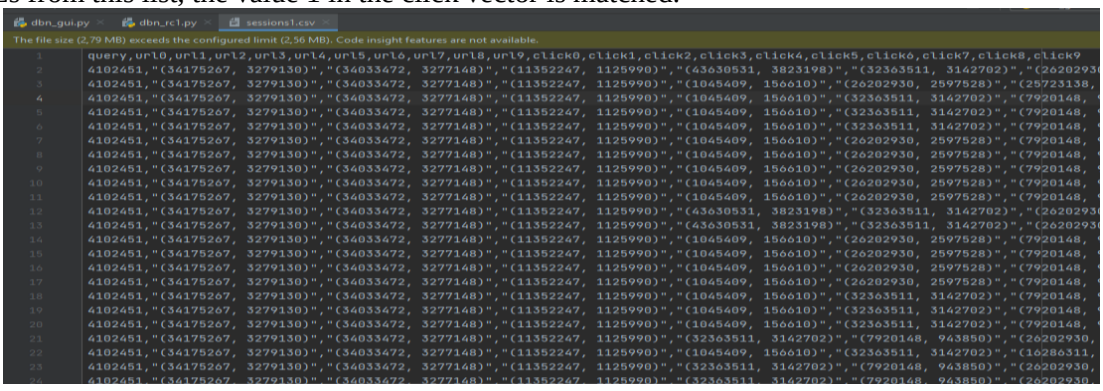


Figure 2: A project for the Flask framework has been created.

For the rest, the value is 0. In this way, a data array is formed, which looks similar to the table. 1. In the file of scripts for parsing logs, the final data can be exported both to the database and to a *.csv file.

4. Results - construction of the Bayesian click model for unbiased CTR prediction.

Model training

A Bayesian network is a directed acyclic graph (Fig. 3). To build it, it is necessary to define a set of vertices (or nodes) and specify connections (arcs, edges) between them. The node *A* from which the edge originates is called the parent, $A = pa(B)$ while the node *B* is called the child. Child nodes

conditionally depend on parent nodes; nodes that have no parents are conditionally independent. Each vertex of the graph corresponds to a random variable that has a finite number of mutually exclusive states. In addition, the probabilities of occurrence of each state are specified.

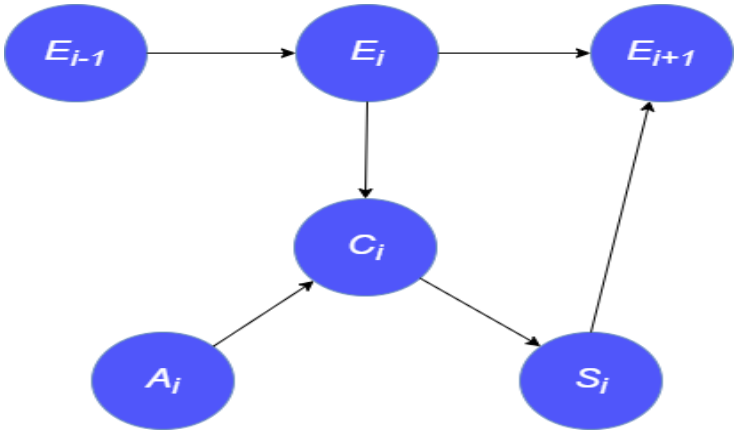


Figure 3. Scheme of the Bayesian click model

The Bayesian click model assumes the existence of the following four binary variables:

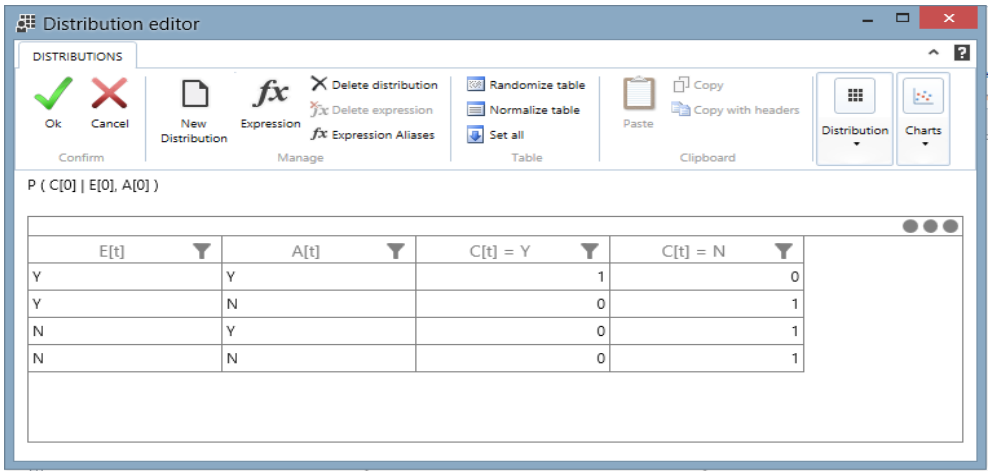
- E – a user viewed some document in the SERP;
- A – a user is interested in a certain document;
- C – a user clicked on this document;
- S – a user was satisfied with the contents of the document.

So the Bayesian network has 4 vertices. Connections between them will be built taking into account the dependencies described by equations (12-15). To implement equations (16-18), the concept of a Bayesian network must be expanded to a Dynamic Bayesian Network (DBN).

DBN is a regular Bayesian network with the concept of time. DBN makes possible to model time series or sequences of dependent events. When modeling clicks in search networks, search sessions with 10 documents in output are considered. Since events E and S are inextricably linked for documents in adjacent positions, a Dynamic Bayesian Network will be used to model clicks within the entire session.

The sequence of DBN construction in Bayes Server consists of the following steps [14]:

1. Creating a node (Node, Discrete Node (temporal)).
2. Naming the node (Name field), specifying the states of the variable ("Y", "N") that correspond to the node.
3. Repeat steps 1-2 for all 4 nodes.
4. Specifying connections between nodes (Links).
5. Setting probability distributions for all states of each variable, using the ratio (12-18) (Fig. 4).



E[t]	A[t]	C[t] = Y	C[t] = N
Y	Y	1	0
Y	N	0	1
N	Y	0	1
N	N	0	1

Figure 4. Probability distribution for variable C. $P(C|pa(C))$

Let's take a closer look at step 4. Since a Dynamic Bayesian Network is not being created, the connections between nodes in Bayes Server have an attribute called temporal order. This attribute is a non-negative integer and in terms of the click model means the following: if the temporal order for the connection from node *C* to node *S* is zero, then *S* at position *i* depends on *C* at the same position. If the temporal order is equal to one, then *S* at position *i* +1 conditionally depends on *C* at position *i*.

Thus, for the construction of the DBN in Bayes Server, additional links with the time order of 1 were created between the variables *S and E* as well as *E and E*. This means, according to formulas (16-18), that the variable *E* for the document at position *i* conditionally depends on the states of variables *E* and *S* at position *i* -1. The final visual view of the dynamic Bayesian network, which was built in Bayes Server, is presented in Fig. 5.

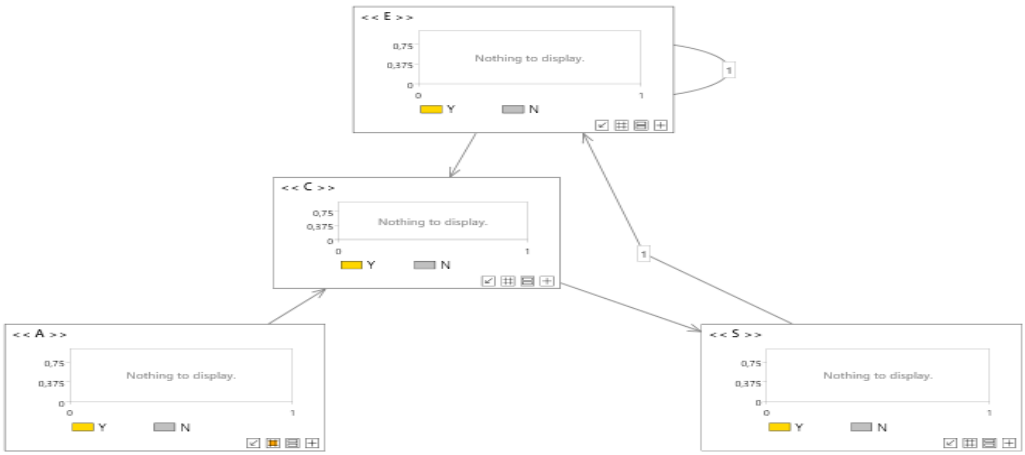


Figure 5. Dynamic Bayesian network (click model), built in Bayes Server

In the PyAgrum library [15], the DBN construction algorithm consists of the following steps:

1. Initialization of an instance of the model class (pyAgrum.BayesNet()).
2. Defining variables (method pyAgrum.LabelizedVariable()), naming and setting states ("Y" and "N", which is equivalent to "Yes" and "No", respectively).
3. Creation of network nodes based on defined variables (pyAgrum.BayesNet().add()).
4. Initialization of directed connections between nodes (pyAgrum.BayesNet().addArc()).
5. Setting probability distributions for the states of each variable (pyAgrum.BayesNet().cpt()).
6. Initialization of the inference mechanism for the model.

In Bayesian networks, inference is the calculation of the values of hidden (or latent) variables based on the values of the observed variables and the probability distribution for the hidden variables. The probabilities of the occurrence of certain states of the variable given *before* research are called *prior probabilities*, when the probabilities calculated *after* the research are called *posterior probabilities*. To determine the posterior probabilities of the latent variables, it is necessary to calculate their conditional and unconditional probabilities. The PyAgrum library provides several algorithms for calculating posteriors, such as Lazy Propagation [16], Variable Elimination, Shafer-Shenoy algorithm, Gibbs sampling, etc. The main algorithm for PyAgrum is Lazy Propagation, so it was used to calculate the DBN hidden variables. It is based on the belief propagation algorithm, which is widely used in machine learning, including for neural networks, Bayesian networks, and Markov models [17].

Lazy Propagation computes all posterior distributions in a Bayesian network using an adjacency tree $T = (C, S)$, where C are cliques of the tree, S are separators. The tree T is built by moralizing and triangulating the graph of the Bayesian network $G = (V, E)$. When the tree is constructed, each cell $C \in C$ is assigned a set of distributions of all variables X such that $\{X\} \cup \{pa(X)\}$. Such a set of distributions corresponding to the cell C is denoted by Φ_C. When evidence $X = x$ is established for some variable X, then from all distributions Φ in the tree, which contain X in the domain of definition, only those for which $X = x$ remain. The algorithm removes the rest.

After the tree *T is initialized*, the message propagation algorithm works. Messages are propagated

from one clique to another through each separator $S \in S$ in two directions - from the leaves to the root of the tree and vice versa. The message $\Phi_{A\to B}$ transmitted from clique A to clique B is a set of probability distributions and is calculated as follows:

$$\Phi_{A\to B} = \sum_{A\setminus B} \Phi_A \cup \mathop{\mathrm{Y}}_{C\in \mathrm{adj(A)}\setminus\{\mathrm{B}\}} \Phi_{C\to A}\,, \tag{22}$$

where $adj(A)$ are cliques adjacent to A. At the same time, variables for which the posterior distribution should not be calculated and for which certificates are not established can be excluded from the calculations.

When traversal of all messages is complete, the posterior distribution for the variable Y can be computed from any clique or separator containing Y. Let Φ be the set of distributions from which the posterior distribution for Y will be calculated. Then the algorithm for calculating $P(Y|\varepsilon)$, where ε is some proof, is as follows:

1. Find relevant distributions R_Y from Φ (Algorithm 3.5, [16]).
2. Exclude R_Y each variable X from the set from $\left\{X \in \mathrm{dom}(\phi) \middle| \phi \in \mathrm{R_y}, X \neq Y\right\}$ (Algorithm 2.3, [16]).
3. Let Φ_Y be the resulting set of distributions. Then calculate $P(Y|\varepsilon)$ as follows:

$$P(Y|\varepsilon) = \frac{\prod_{\phi\in\Phi_Y}\phi}{\sum_Y \prod_{\phi\in\Phi_Y}\phi} \tag{23}$$

In PyArgum, in order to specify the inference method for the newly created Bayesian network, it is only necessary to initialize an instance of the pyArgum.LazyPropagation(bn) class, where bn is an instance of the pyAgrum.BayesNet() class.

After defining the structure of the model, PyArgum provides an opportunity to generate a visualization of the Dynamic Bayesian Network in svg format. Fig. 6 shows the model created by means of the library.

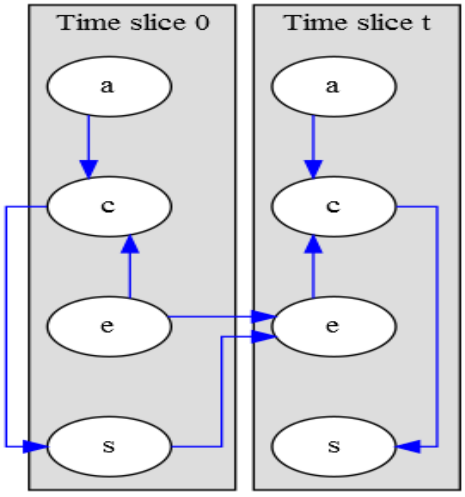


Figure 6. The model was created using the tools of the PyAgrum library

After constructing the DBN, both in Bayes Server and in Python, posterior probabilities were found for the variables A_i and S_i, where i are time points $i = \overline{0,9}$ (for DBN, the numbering of time points starts from zero). For this, the variables C_i were defined as observables and the states, $C_i =$"N", were defined for all time slices $i = \overline{0,9}$. In Bayes Server, the states of the observed variables must be specified in special tables, while in PyAgrum there are two ways to initialize variables: using setEvidence() or pyAgrum.lib.dynamicBN.plotFollow(). The plotFollow() method

immediately generates a plot of the given variables, so this one was chosen for use in the inference. In PyAgrum, the inference method is Lazy Propagation, in Bayes Server – Relevance Tree. Both of these algorithms belong to the family of exact inference algorithms, but the developers of Bayes Server do not reveal details about how their algorithm works. After configuring the models, we perform calculations. The results of the calculations are shown in figures 7-8. It is worth noting that for these cases $a_u = s_u = 0.5$

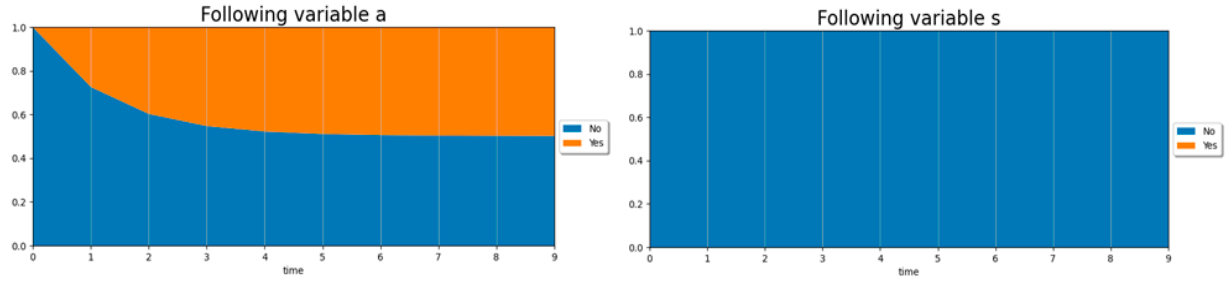


Figure 7. Visualization of posterior distributions for A_i and S_i in PyAgrum

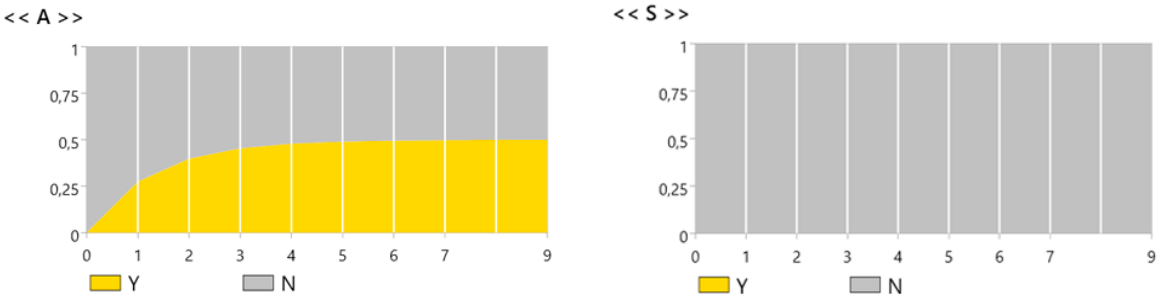


Figure 8. Visualization of posterior distributions for A_i and S_i in Bayes Server

From figures 7 – 8 shows that the calculation results are identical, despite the different methods of inference. Therefore, the tool for building a DBN for the purpose of CTR prediction can be both the Bayes Server API, which is available as Python libraries, and PyAgrum directly. Empirically, it was established that the calculation of posterior probabilities in PyAgrum is faster, so Bayes Server will remain only a secondary tool in the study of Bayesian networks.

To predict CTR, it is necessary to calculate the parameter a_u, which is initialized at the request level. That is, it is determined only by the request-document pair. The parameter a_u is equivalent to the CTR of document u if it were in the first position. Therefore, the value a_u will be the unbiased CTR prediction for u

To calculate the parameters a_u and s_u, a two-step iterative Expectation-Maximization algorithm [10] was used.

Suppose that for a fixed search query q are investigated n search sessions. Let A_i^j, S_i^j, E_i^j are hidden variables at i position, $i = \overline{1,10}$ in j search session, $j = \overline{1,n}$. We denote the document at position i for session j as d_i^j. Then one iteration of the algorithm consists of two steps.

Step 1. **Maximization.** Given $Q(A_i^j)$ and $Q(S_i^j)$ posterior distributions, then the updated parameters a_u, s_u will be found as follows:

$$a_u = \arg\max_a \sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u)\left(Q(A_i^j = 0)\ln(1-a) + Q(A_i^j = 1)\ln(a)\right) + \ln P(a);$$

$$s_u = \arg\max_s \sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u, S_i^j = 1)\left(Q(AS_i^j = 0)\ln(1-s) + Q(S_i^j = 1)\ln(s)\right) + \ln P(s),$$

where I is the indicator function, P (a) and P (s) are a priori beta distributions taken with parameters (1, 1).

The maximum a posteriori at this step can be calculated in two ways: by numerical optimization (gradient descent method, Newton's method) or analytically (in closed form). For numerical optimization, the optimize module of the scipy library has a special minimize_scalar() method. To calculate a_u, s_u the theory of conjugate distributions is used analytically. We denote α_a, and β_a as hyperparameters for a_u, α_s and β_s as hyperparameters for s_u. Then updated a_u and s_u will be calculated as follows:

$$a_u = \frac{1+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u)\left(Q(A_i^j = 1)\right)}{2+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u)\left(Q(A_i^j = 1)\right)+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u)\left(Q(A_i^j = 0)\right)};$$

$$s_u = \frac{1+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u, S_i^j = 1)\left(Q(S_i^j = 1)\right)}{2+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u, S_i^j = 1)\left(Q(S_i^j = 1)\right)+\sum_{j=1}^{n}\sum_{i=1}^{10} I(d_i^j = u, S_i^j = 1)\left(Q(S_i^j = 0)\right)}.$$

Since closed-form computation requires less programming time and is more accurate, this method was chosen to perform the maximization step.

Step 2. **Expectation.** a_u, s_u become the new parameters for A and S, respectively. Then new posterior distributions $Q(A_i^j)$ and $Q(S_i^j)$ are calculated.

Iterations are performed until the parameters a_u, s_u converge.

Creating a model in Python was implemented in the model_setup function with input parameters initial values a_u and s_u input parameter values γ.

Model training in Python was implemented in the train function, which accepts the following input parameters:

- model – DBN created using PyAgrum;
- sessions_number – the number of training sessions for the model;
- df_with_clicks is a Pandas dataframe that contains click logs in the format shown in the Table 1;
- max_iterations (optional) – the number of iterations of the EM algorithm. (max_iterations=60).

After the training process is complete, the function returns matrices a and s that contain the corresponding parameters for each session and each URL in the session.

A graphical user interface (GUI) was also created for the program that implements the Bayesian click model to combine input and output information for the model. The visual shell is presented in fig. 9:

Figure 9. GUI for DBN

To start the learning process, you need to set the following input data:

- Parameter γ is an input parameter of the Bayesian click model. γ is the probability that the user will continue the search if the previously viewed document was not relevant for him;
- Training data path is a folder in which click logs are stored, for which there is only one unique search query within one file;
- Search query amount defines some sample of data that will be used to train the model;
- Documents: "all unique URLs" if a_u calculated for all documents on the search page. If it is necessary to calculate a_u only for a specific list of documents, then you should select the item "URLs from a config file";
- Number of training sessions can be specified as a single number or a sequence of unique integers. In the second case, training will be carried out sequentially in several stages with different numbers of training sessions in each;

During model training, the calculated parameters are recorded in the corresponding table. By default, it has only two columns - "Query" and "Document". During the execution of the program, the table changes dynamically after activating the "Evaluate" button. In addition to calculating and displaying the results in a table, it is also possible to save these results in a separate file, in .csv format or as a database. There is an option to save only the values a_u as a matrix, without the corresponding URLs and queries.

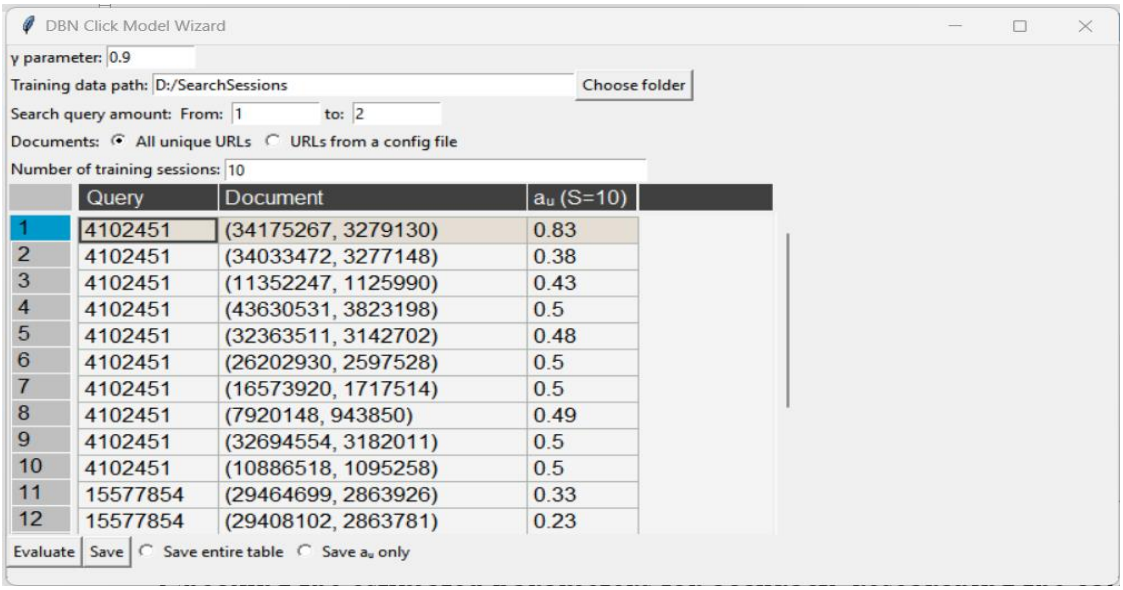


Figure 10. GUI for DBN (results of learning)

Checking the estimated parameters for accuracy and research of the calculation error

Since a_u and s_u are evaluated by an iterative algorithm, it is necessary to study the stopping criterion of this algorithm. For this purpose, an experiment was conducted, which consists of the following stages:

1. Randomly select 5 URLs associated with different search queries.
2. Set a fixed number of iterations for the EM algorithm (max_iterations = 100).
3. Compute values a_u for all five URLs over a fixed number of training sessions for a given number of iterations. At the same time, for each URL, make two stages of calculation a_u: with initial values of 0.1 and 0.9. In addition, fix all intermediate values a_u that are calculated at each iteration.
4. Plot convergence graphs a_u for different numbers of training sessions.

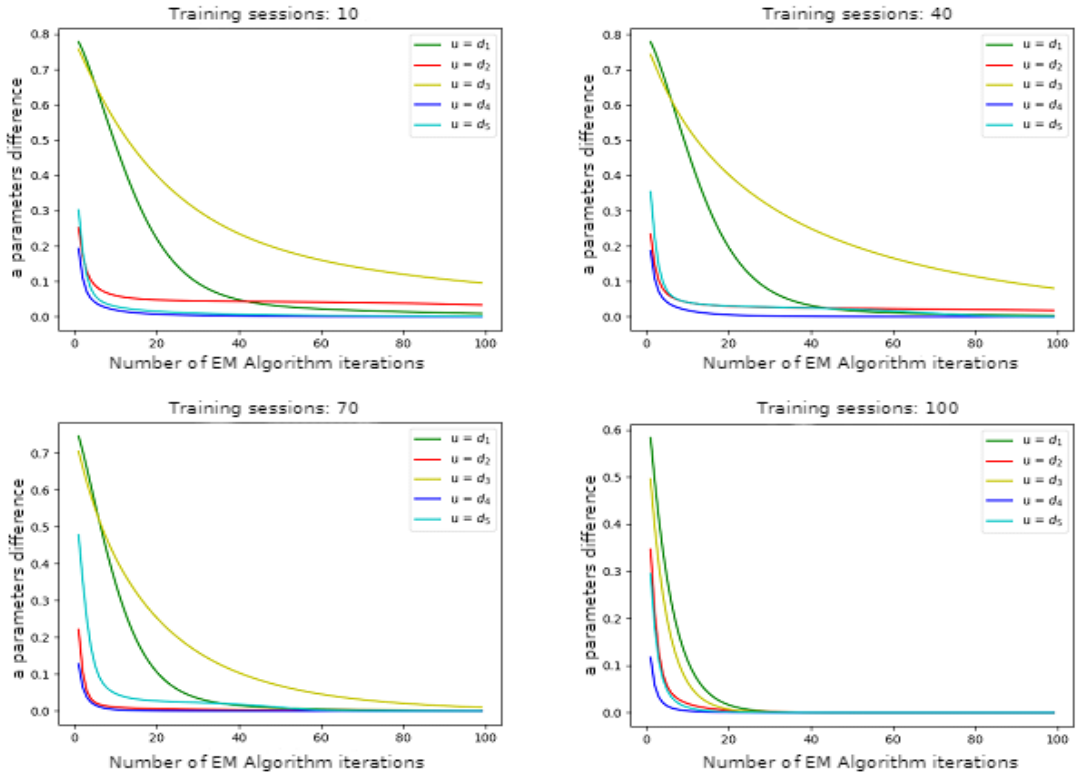


Figure 11. Graphs of dependences of differences a_u on the number of iterations of the EM algorithm

Figure 11 shows that with an increase in the number of training sessions, the initial values a_u have less and less influence on their final evaluation. For four out of five documents, the change in differences slows down with the number of iterations exceeding 60. For $u = d_3$ a similar result is achieved only on a large number of sessions. For further experiments, the maximum number of iterations of the EM algorithm will be 60.

As previously stated, a_u as a measure of the attractiveness of some URL u is equivalent to the CTR u, if it were in the first position in the output. Therefore, to study the accuracy of CTR prediction by the Bayesian model, an experiment was conducted according to the protocol originally proposed in [12], which was also used in [18]. The experiment consists of the following steps:

1. View all sessions that share a search query.
2. Consider some URL that appeared in both position 1 and positions 2-10.
3. All sessions, where the URL appeared in position 1, are considered test sessions.
4. The rest of the sessions are considered training sessions.
5. In the test sessions, calculate the CTR for the URL directly according to the formula.
6. In training sessions, train the model and evaluate the parameter a_u.
7. Compare the test CTR with a_u, calculating the error.
8. Average the errors for all similar request-document pairs, weighting them by the number of test sessions. Thus, the weighted root mean square error for this experiment was calculated according to the following formula:

$$MSE = \sum_{i=1}^{n} \frac{w_i (CTR_i - a_{ui})^2}{\sum w_i}$$

As part of this experiment, averaging was performed on different numbers of request-document pairs (10, 20, 30, 40, 50) for a better understanding of the dynamics of error changes. The results are shown in the graphs in fig. 12.

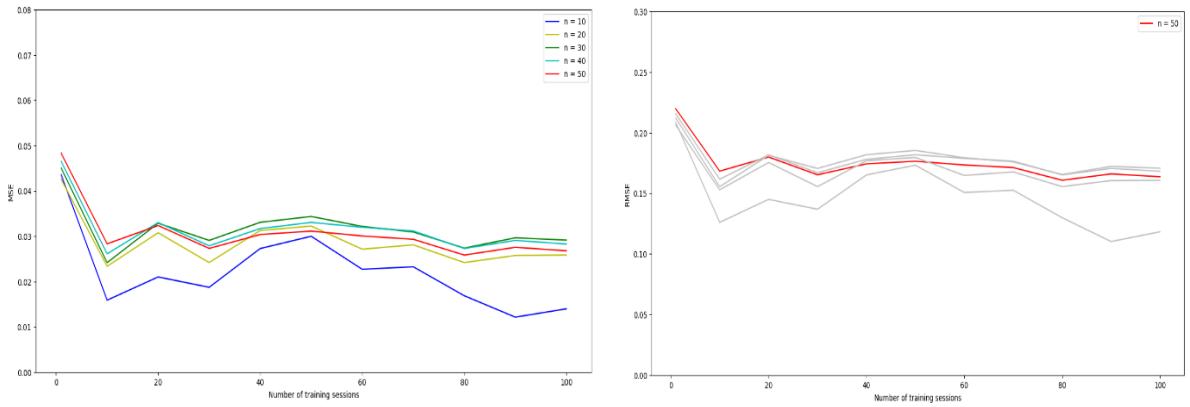


Figure 12. Graphs of the dependence of errors on the number of training sessions

After comparing the graphs in fig. 12, it can be concluded that with an increase in the number of request-document pairs for averaging, the graphs become smoother - the change in error becomes smaller with an increase in the number of studied URLs. In addition, there is a downward trend, that is, with an increase in the number of sessions, the root mean square error decreases, which is quite natural. On the right of fig. 12 are the graphs of the change in the value, which is the square root of the mean square error (RMSE). Thus, it is possible to see what is the average deviation of the predicted CTR from the real one, depending on the number of training sessions.

It is also important to remember that this click model requires the definition of an input parameter γ, which is the probability that the user will continue the search, given that the previous result did not satisfy him. Thus, the parameters γ also depend on the estimates a_u and s_u. For previous experiments, γ it was equal to 0.9. To find the optimal value of this parameter, an experiment similar to the previous one was conducted, but now the number of training sessions and *n* are fixed, and the error is presented as a function dependent on the input parameter γ (fig. 13).

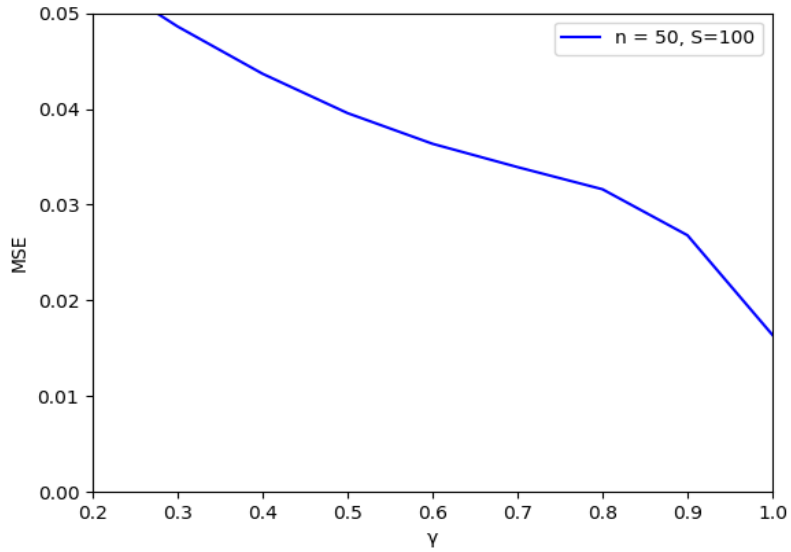


Figure 13. The dependence of the root mean square error on γ

An unexpected conclusion emerges from the results of the experiment - the best CTR forecast is achieved at the value γ =1, which corresponds to the SDBN (Simplified DBN) specification. A similar effect was also observed in [18]. In other words, for the studied click log, SDBN predicts CTR better than general DBN. This means that users, according to these click logs, are extremely persistent in their search for information. Since the best prediction is achieved for γ =1, let's conduct an experiment with the calculation of the root mean square weighted error at γ =1 and compare it with the error at γ =0.9.

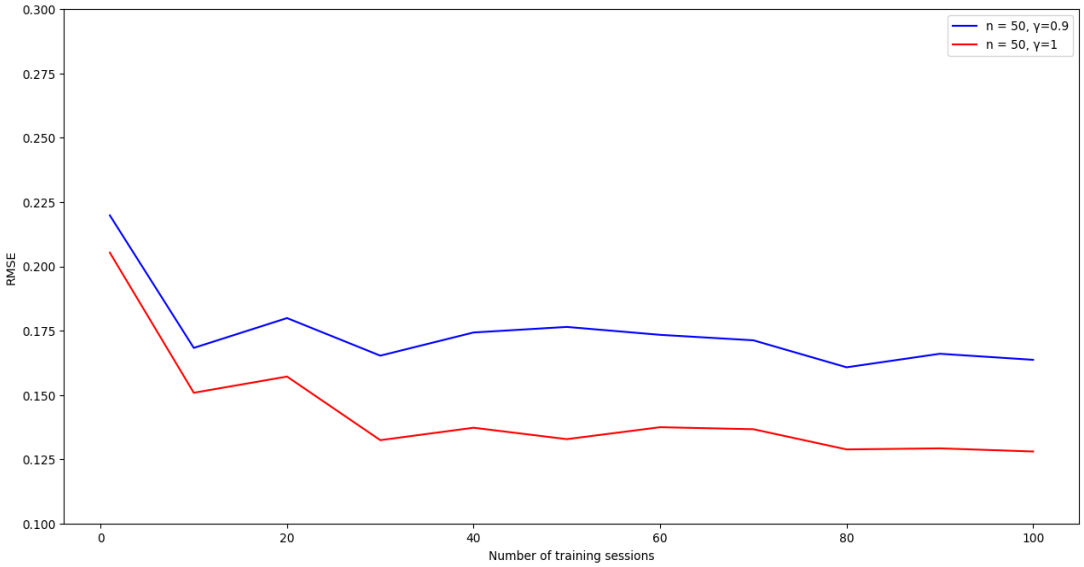


Figure 14. The root mean square error. Comparison for γ =0.9 and γ =1

Comparing forecasts for two different values γ, we come to the conclusion that the optimal DBN parameter for predicting CTR on these click logs is γ =1, and for any number of training sessions (fig. 14).

5. Conclusion

1. The Bayesian click model is considered, including probabilistic relationships that describe the model, semantics of model parameters, possible methods of parameter estimation.
2. An overview of available software products and services for analyzing and predicting user click behavior was conducted.
3. With the use of modern information technologies and programming languages, in particular, the Python programming language, the PyAgrum library, and the Bayes Server software, a click model was built.
4. A comparison of the performance results of various algorithms for calculating hidden variables was carried out. An application in the Python programming language was created, which evaluates unknown model parameters based on click logs.
5. Studies of the accuracy of CTR prediction by the click model have been conducted. In particular, the dependence of the prediction error on the number of training sessions, as well as on the value of the input parameter, is illustrated. By conducting the experiment, the optimal value of the input parameter was found.

In future studies, it is planned to develop a Bayesian click model for medical calculators for cardiac diagnostics [19, 20], biosensor systems [21, 22] and geoinformation systems [23]. This approach will allow taking into account probabilistic ratios that will describe the proposed models, the semantics of parameters, as well as possible methods of their evaluation.

6. References

[1] Y. Yang, P. Zhai. Click-Through Rate Prediction in Online Advertising: A Literature Review. SSRN Electronic Journal. 2022.

[2] M. Khvostivskyy, H. Osukhivska, L. Khvostivska, T. Lobur, D. Velychko, S. Lupenko, T. Hovorushchenko, Mathematical modelling of daily computer network traffic. ITTAP 2021. CEUR Workshop Proceedings. Ternopil, Ukraine, November 16-18, 2021. Vol. 3039. pp.107-111.

[3] V. Zhukovskyy, S. Shatnyi, N. Zhukovska, A. Sverstiuk, Neural network clustering technology for cartographic images recognition. EUROCON 2021 - 19th IEEE International

Conference on Smart Technologies, Proceedings, 2021, pp. 125–128. doi: 10.1109/EUROCON52738.2021.9535544.
[4] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, Deep interest network for click-through rate prediction, in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2018, pp. 1059–1068.
[5] H. Zhang, J. Yan, and Y. Zhang, CTR prediction models considering the dynamics of user interest, IEEE Access, vol. 8, 2020. pp. 72847–72858.
[6] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, Deep interest evolution network for click-through rate prediction, in Proc. AAAI Conf. Artif. Intel., vol. 33, 2019, pp. 5941–5948.
[7] Z. Xiao, L. Yang, W. Jiang, Y. Wei, Y. Hu, and H. Wang, ``Deep multiinterest network for click-through rate prediction," in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2020, pp. 2265–2268.
[8] X. Li, C. Wang, B. Tong, J. Tan, X. Zeng, and T. Zhuang, ``Deep timeaware item evolution network for click-through rate prediction," in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2020, pp. 785–794.
[9] Y. Feng, F. Lv, B. Hu, F. Sun, K. Kuang, Y. Liu, Q. Liu, and W. Ou, "MTBRN: Multiplex target-behavior relation enhanced network for clickthrough rate prediction," ' in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2020, pp. 2421–2428
[10] Q. Pi, W. Bian, G. Zhou, X. Zhu, and K. Gai, Practice on long sequential user behavior modeling for click-through rate prediction, in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 2671–2679
[11] A. Chuklin Click Models for Web Search / A. Chuklin, I. Markov, M. Rijke. // Synthesis Lectures on Information Concepts, Retrieval, and Services.. - 2015. - No. 3. – pp. 1–115.
[12] O. Chapelle A Dynamic Bayesian Network Click Model for Web Search / O. Chapelle, Y. Zhang. – 2009. – pp. 1–10.
[13] T. Krishnan, G. McLachlan. The EM algorithm. In Handbook of computational statistics / - 2012. pp. 139-172.
[14] The official site of Bayes Server [Electronic resource] - Access mode: https://www.bayesserver.com/ .
[15] PyAgrum official site [Electronic resource] - Access mode: https://pyagrum.readthedocs.io/en/1.1.1/ .
[16] A. Madsen, F. Jensen Lazy propagation: A junction tree inference algorithm based on lazy evaluation. Artificial Intelligence. 1999. No. 1-2. pp. 203–245.
[17] A. Madsen, C. Butz. Exploiting Semantics in Bayesian Network Inference Using Lazy Propagation. 2015. pp. 3–15.
[18] A. Grotov Comparative Study of Click Models for Web Search / A. Grotov et al. // Lecture Notes in Computer Science. Cham. 2015. pp. 78–90.
[19] S. Lupenko, I. Lytvynenko, A. Sverstiuk, A. Horkunenko, B. Shelestovskyi, Software for statistical processing and modeling of a set of synchronously registered cardio signals of different physical nature. CEUR Workshop Proceedings, 2021, 2864, pp. 194–205.
[20] V. Martsenyuk, A. Sverstiuk, A. Klos-Witkowska, A. Horkunenko, S. Rajba, Vector of diagnostic features in the form of decomposition coefficients of statistical estimates using a cyclic random process model of cardiosignal. Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019, 1, pp. 298–303. doi: 10.1109/IDAACS.2019.8924398.
[21] V. Martsenyuk, A. Sverstiuk, I. Gvozdetska, Using Differential Equations with Time Delay on a Hexagonal Lattice for Modeling Immunosensors. Cybernetics and Systems Analysis, 2019, 55(4), pp. 625–637. doi: 10.1007/s10559-019-00171-2.
[22] V. Martsenyuk, A. Klos-Witkowska, A. Sverstiuk, Stability Investigation of Biosensor Model Based on Finite Lattice Difference Equations. Springer Proceedings in Mathematics and Statistics, 2020, 312, pp. 297–321. doi: 10.1007/978-3-030-35502-9_13.
[23] T. Vilkys, V. Rudzinskas, O. Prentkovskis, N. Višniakov, P. Maruschak, Evaluation of failure pressure for gas pipelines with combined defects, 2018. Metals8(5), pp. 346.