information

Article

Reinforcement Learning Page Prediction for Hierarchically Ordered Municipal Websites

Petri Puustinen [1,*], Kostas Stefanidis [1,2], Jaana Kekäläinen [1,2] and Marko Junkkari [1,2]

[1] Information Technology and Communication Sciences (ITC), Tampere University, Kalevantie 4, 33100 Tampere, Finland; konstantinos.stefanidis@tuni.fi (K.S.); jaana.kekalainen@tuni.fi (J.K.); marko.junkkari@tuni.fi (M.J.)
[2] Faculty of Information Technology and Communication Sciences (ITC), Tampere University, Kalevantie 4, 33100 Tampere, Finland
* Correspondence: petri.puustinen@tuni.fi

Abstract: Public websites offer information on a variety of topics and services and are accessed by users with varying skills to browse the kind of electronic document repositories. However, the complex website structure and diversity of web browsing behavior create a challenging task for click prediction. This paper presents the results of a novel reinforcement learning approach to model user browsing patterns in a hierarchically ordered municipal website. We study how accurate predictor the browsing history is, when the target pages are not immediate next pages pointed by hyperlinks, but appear a number of levels down the hierarchy. We compare traditional type of baseline classifiers' performance against our reinforcement learning-based training algorithm.

Keywords: clickstream analysis; markov model; deep learning; reinforcement learning; Q-learning; hierarchically ordered website

Citation: Puustinen, P.; Stefanidis, K.; Kekäläinen, J.; Junkkari, M. Reinforcement Learning Page Prediction for Hierarchically Ordered Municipal Websites. *Information* **2021**, *12*, 231. https://doi.org/10.3390/info12060231

Academic Editor: Symeon Papadopoulos

Received: 21 April 2021
Accepted: 26 May 2021
Published: 28 May 2021

1. Introduction

Public domain web portals, for example, municipal websites, education, or healthcare services, are often organized into a hierarchy. The hierarchy represents a tree with a single root node (the main home page) with several branches representing different topic pages, i.e., each topic has its own branch starting from the root. When the content of a web page grows large, the information organization of the portals may not be intuitive for all users. They may find it laborious to reach the information they need or, in the worst case, do not find it at all. The user experience is poor if users are engaged in browsing activities where finding the relevant information is compromised. Recommendation of pages is a solution to minimize the user effort to find relevant information in a large information space. It is widely used in commercial websites but more sparsely in public domain portals.

Table 1 shows the simple URL paths of four target page examples. We see the page structure that is encoded in the path segments, where each topic and level of information are seen separated by a forward slash character. Each segment corresponds to a web page connected via a hyperlink to each following segment or a target page. The whole topology appears as a directed graph, where the main page is the root node following each main topic in their own child branch spanning in their own branches in every subtopic. There is usually a generic-specific type of relation between any given parent-child node, where traversal to the lower levels of the hierarchy accesses the more specific and desired content in that branch of a topic.

Users' information seeking activity involves traversal via intermediate pages that serve as a guide towards the most specific information in the leaf pages in the hierarchy until relevant information is found. The complexities' contributing to a perceived user experience may include the layout of the portal, wording and conceptualization intermediate subtopics, and the number of levels at the intermediates.

Item recommendation is commonly used as an aid for users to find a candidate items of interest. The methods by large extend are based on users' past interactions to model similarities as weights between adjacent items on a website. A shortcoming for the item similarity approach is a short-term consideration for the importance of items. Inevitably, if we seek to recommend target pages, it is not feasible to recommend intermediate pages which are used mostly for navigation purposes. Thus, single step user-to-item or item-to-item matching is not seen as a useful option, but to provide means for recommendation that extends multiple levels down the branch structure.

Table 1. Path examples from an example municipal website.

Example	Path
1	<root>/philharmonic/concerts/first-of-may.html
2	<root>/health-services/dental-treatment/dental-clinics.html
3	<root>/sports-and-leisure/recreation/outdoor-activities/summer-camps.html
4	<root>/sports-and-leisure/recreation/outdoor-activities/ski-tracks.html

Additionally, it is not uncommon that the explicit preference of user is not known. In other words, they are treated as anonymous for usability reasons and for privacy concerns. In that case, when longer-term user history or profile is not present, there may be limitations how the probability of related items can be calculated. However, the recommendations can adapt the short-term interest of an anonymous user in session-based recommendation setting. The adaptation is the bound to the current situation or short-term interests. This is known as contextual awareness, including representational and interactional factors, where the former may use features, like geographical information or time, and the latter, user behavioral patterns [1].

Traditional recommendation techniques, such as collaborative filtering (CF), are based on users' expressions on similar items [2–4]. In broad terms, the CF method may use a matrix completion setup where similarity is expressed as ratings of items for user-item similarity. Analyzing the content of textual information to find regularities is the core of content-based filtering techniques for item-item setup [3]. In sequence-aware recommender systems, the assumption is that the underlying data is a sequence dataset consisting of user actions in a user community, where the actions may have additional attributes, such as item view, a click event, or timestamp [1].

We focus on the users' click event history and model their behavior using the events as a single feature to study the applicability of traditional item-based classifier algorithms and our own target-based approach to a hierarchical content domain. The past click events record the paths from the upper levels of the hierarchy to the final content pages. Based on the recorded history of click events on a website, the task is to map the previous events x to a related event y. A common model of the mapping is to calculate the probability $p(y|x)$ and $p(y_1, ..., y_k|x_1, ..., x_k)$ of all clicks in a k length session [5].

Our intention is to observe how the properties of the document hierarchy affect the performance in a sequence-aware recommender system scenario of a baseline, deep learning (DL) and reinforcement learning (RL) type of classifiers. We choose traditional supervised learning approaches to compare their applicability for future step prediction. A Markov Model (MM) is used to provide a naïve model baseline for the specific used data. It has been used traditionally for WNP, although it is known to fail to predict more than one step ahead [6]. MM is also used to compare the results of the DL implementations used in this study for the scale of the performance. *Long short-term memory* (LSTM) models are designed to model the temporal dependencies of random variables. It is a natural choice for the path prediction. In addition, a *multi-layer perceptron* (MLP) is chosen as an option for its simplicity to compare the generic performance of the models.

RL differs from supervised learning by its characteristics of mapping situations into actions. An agent learns by taking actions on trial and error search which yields reward depending on the goodness of given action. The agent's ability to learn complex scenar-

ios depends not only on immediate (labeled) rewards but the ability to evaluate which subsequent rewards lead to a desired goal. The maximization of a total reward may then be delayed due to the collected set of subsequent rewards [7]. It is, therefore, suitable for multi-step goals and if the targets are set as a leaf pages in the page hierarchy.

We present RL-based Q-learning approach for modeling page recommendations for a hierarchical website using clickstreams as our training data. We present the modeling results of a supervised DL approach for page recommendations. We train three types of classifiers to predict the probability of the next click event based on users' browsing history. We will use accuracy to evaluate the classifiers given a random click and a subsequent event from the historical events in relation to the classifiers' output. We use the municipal website from the Finnish city of *Tampere* (www.tampere.fi (accessed on 21 April 2021)) as our case study.

The contributions of this paper are:

- We evaluate the accuracy of the classifiers given a single random page.
- We introduce *session-trajectory learning*; a novel Q-learning algorithm designed for training in hierarchical web domain to predict final content pages.
- We observe the effect of history of click events in relation to the subsequent event to the classifiers' performance.
- We experiment the effect on the performance of different deep learning architectures and compare then with our algorithm.
- We evaluate the applicability of the sequential models for the domain and data.

The rest of this paper is organized as follows: Section 2 presents the related work by describing the aspects of recommender systems. Section 3 introduces the modeling, the baseline methods, and the designed algorithm used in this work. Section 4 describes the municipal website data and the evaluation criteria used for comparing the models. Section 5 provides the results of the model comparison. Section 6 discusses the result analysis of the experiments. Finally, Section 7 includes the conclusions and future work of the study.

2. Related Work

The evaluation of item popularity is the key element for CF, where explicit users' ratings may be used identify similar items or the use of the amount of clicks for further ranking [3,4], where popularity bias [8–10] can introduce unfairness to it. User browsing patterns are usually extracted from web logs for WNP [11], which are commonly used in other areas, such as Information Systems research for user engagement studies [12]. Whereas WNP addressed the problem with predicting subsequent click events and is by nature a multi-class problem, it shares similar popularity-based heuristics as CF. Technically, the assessment of the output may use probability threshold or top-N of most probable outcomes heuristics [11]. In sequence-aware recommender systems, the input is a sequence of user actions in a user community and the output, a sequence of ordered list of items. Sequence-aware recommender systems can be categorized in application scenarios, such as context adaptation, trend detection, repeated recommendation, or order constraints and sequential patterns [1].

MM have been traditionally used for WNP [6] as for contemporary designs [11], even though its limitations to predict more than one step ahead [6]. Only 1st order MM is commonly used due to the growing state space complexity [6]. A DL-based recommendation is a popular research topic [13] and have been found to be an alternative for matrix factorization as they are able to generalize functions well [14]. Typical e-commerce environment use cases using LSTM, for example, includes predicting a buy event [5] and ad clicks [15].

DL methods are also popular for their ability to extend the method feature space. The categorical value embedding has been found to help DL to generalize better for sparse data with unknown statistics [16]. The feature interactions have also been learned through a factorization machine-based DL architectures [17,18]. Wide & Deep learning integrates continuous and categorical features in app recommendation for the Google Play app

store [19]. The solution uses wide linear models with the deep neural networks to combine the memorization (learning frequent item co-occurrences) and generalization (correlation transitivity and different feature combinations) to also overcome the sparsity and over specialization problems.

In recent years, RL-based models have been applied to graph generation solutions. Moreover, Graph representation learning (GRL) is a solution to learn low-dimension latent representations of graph topology. Methods include random walk and matrix factorization-based methods, as well as graph neural networks (GNNs) [20]. Use cases, such as link prediction in a social network, recommendation in e-commerce, multi-label classification, or language modeling, have been successfully used methods, such as DeepWalk [21], Node2Vec [22], or APP [23]. The methods can be broadly described as methods learning latent representations of graphs using a random walk procedure, where the walks are input to neural models, such as word2vec.

A generic Q-learning model approaches directed acyclic graph (DAG) generation as actions corresponding to the addition of nodes and edges. A directed graph $G = (V, E)$ is represented as a binary adjacency matrix A with an additional feature matrix F representing a type for each node. A positive reward is given if agent produces a DAG isomorphic to a ground truth graph. Ref [24] uses a Graph Convolutional Policy Network (GCPN) for molecular structure generation leverages convolutional network-based RL model [25]. It learns molecular structures modeled as adjacency matrix A, feature matrix F and E edge-connected adjacency tensor in a graph $G = (A, E, F)$. Its goal is to form a structure, that resembles realistic molecules. Hierarchical Reinforcement Learning is used to learn chains of reasoning paths over a Knowledge Graph (KG) [26]. It shows the ability to learn the relation between the head and the target entities of the given query against KG relationships, as well as predicting the target entity given head and a relation.

Reinforcement Learning has been applied to e-commerce recommendation as Q-learning problem using a common setting of log data of users' past visits to probable other pages of interest [27]. Relating buyers' impression to a potential seller via buyer used search words [28]. Similar use cases of relating customer feedback to items through Q-learning and self-attention mechanism [29]. Graph scenarios have been approached using a knowledge graph to discover high-quality negative signals from implicit feedback to interpret user intent [30]. Graph Convolutional Q-network (GCQN) learns recommendation policies based on the inputs of the graph-structured representations [31]. A sequence type of scenarios have been investigated with previous enrolled course relation to most relevant next courses [32]. The closest domain of a use is the use of hierarchical clustering built over items, where a path to a certain target item is formulated [33].

A sequence-based recommender is a known solution domain [1]. However, the type of municipal website and the hierarchical taxonomy, to the best of our knowledge, has not been addressed in research community [1,3,4,13,34–37], where we wish to predict the target pages of a chain [38] type of browsing behavior following a link structure via intermediate pages. There are similar use cases in both graph generating solution and recent recommender research using RL and Q-learning that be drawn inspiration from for path prediction over usage graphs.

3. Modeling and Methods

In this section, we formulate the problem and a model for the history of users' click events to be used for page prediction. We then explain how the baseline classifiers use the history to model the probability of subsequent click events. The last part describes our approach to modeling a hierarchical web page usage graph through Q-learning and how the target pages are treated in comparison with the intermediate pages. The main notations used in this paper are summarized at Table 2.

Table 2. Main notations.

Symbols	Meaning
E	Sequence of click events
E^*	Set of all possible sequences of E
o	Order
d	Distance
I	Set of items
C	Set of users
L	Ordered list of items of length n
L^*	Set of all possible lists L of length up to n
$util(c, L)$	Utility function, where $c \in C$ and $L \in L^*$
U	Set of web page URLs
$prob(x)$	Probability function of click events given an input sequence
$G = (V, E)$	Constructed web graph, where $V \subset U$
$Q(S, A)$	Estimation policy

3.1. The Problem Definition

The domain of our research is a municipal website, where the content is organized as a tree hierarchy where the main page represents a single root node. Each topic continues from the root node in its own branch. Each of the subsequent child node represents a page used in navigational purpose. These intermediate pages will eventually lead to a leaf page that contains specific content. Users' seek information by traversing via a number of these intermediate pages towards the most specific information in the leaf pages of the hierarchy.

The users are treated as anonymous, i.e., their preferences cannot be characterized using an implicit profile. Therefore, we observe and adapt to the session-based features referred as representational and interactional factors. In this study, we concentrate on the latter which consists of behavioral patterns which includes the click event sequences of the user community that are used to model trajectories leading to target pages.

A collection of user actions and requests in a website visit are called a *server session*. *Clickstream* is a list of web pages visited in the session presented in the viewing order [39]. It is hereby defined as a sequence $(E_i)_{i=1}^{k} = (e_1, e_2, ..., e_k)$ of k click events e initiated by a single *active user*. E^* is a domain of all clickstream *sequences* in the database.

The current page, a click event e at time t, of a user is defined as a context page e_{ctx}. *Order* o refers to how many previous $t - o$ pages the next page depends on adopting the formalism from Markov modeling [40]. *Distance* d represents the number of subsequent future pages $t + d$ in relation with the e_{ctx}. The overall context of an active user is the sequence observed in the current session, where the e_{ctx} is the main observation of the short-term interest. A target leaf page is defined as e_{tgt} which in the tree hierarchy includes the set pages that do not have any child pages. Our goal is to predict the leaf content given that an active user e_{ctx} is any of the intermediate pages, including the root and for any number $\forall d$ of intermediate pages between e_{ctx} and the e_{tgt}.

Let I be the set of all recommendable items on the website and C a set of all users. L is an ordered list of items of length n, where each sequence L is an element of the set of the powerset of I, which is denoted as L^*, i.e., a set of all permutations of the power set. We want to design a utility function $util$, which purpose is to find an ordered list of items L of length n for user $c \in C$. The purpose of the sequence-aware recommendation problem is to determine a sequence L_c for a user that maximizes the score of a given sequence. The design of the function depends on what type of value the recommender should provide to the user [1].

$$L_c = \arg\max_{L \in L^*} util(c, L). \tag{1}$$

Our purpose is to find a function that predicts the leaf content by providing a sequence of pages having a target page as a last item. The solution aims to use the clickstream records of past user sessions to model paths to the leaf pages. It needs to figure out a weighting scheme that describes the paths in a such way that are agnostic to the underlying page

structure but generalize the sequential regularities presented in a user community to describe a policy $\pi : e_{ctx} \rightarrow e_{tgt}$ from any given e_{ctx}.

Given such a kind of scenario, we can model the problem using Q-learning. It is designed to maximize the expected value of the total reward over successive steps towards a target. However, we need to design a training algorithm to use clickstream data to model hierarchical page topologies and usage, which is not limited to any given topology per se, but can be generalized any kind of structures as long as the structure is represented in the input data. We compare the performance in a sequence-aware recommender scenario of our RL-based implementation against the MM and DL type of baseline classifiers that are derived from traditional recommender methods used in cases, such as a buy event [5] or an ad click [15] prediction.

We limit each item on I to be a web page in our given page hierarchy, in which all have a designated URL, from which leaf page information can be derived, e.g., a page having no child nodes what was described in simple examples in Table 1. U is a set of URLs of the web pages available on the website $u_i \in U$. $|U| = m$ is the number of available URLs defining the state space of the classifier, i.e., the space is the equal amount of input and output the neurons of a classifier and it relates to the space and computing requirements for the training. A precondition for the method usage is that each event is encoded as a unique integer value $(u_i \subset \mathbb{N}_1)$. Any subset of clickstreams are denoted by E_z, where z represents a label for the selection criteria.

The modeling uses a subset $E_z = \{z_1, z_2, ..., z_h\}$ of all dataset sequences $(E_z \subset E^*)$, where the aim is to use a smaller number of training samples that are available at the whole dataset. Each sequence z_j length has an upper and lower bounds $\{z_j | z_j \in E_z, |z_j| \geq Len_{low}, and\ |z_j| \leq Len_{high}\}$, where the lower bound is dictated by $Len_{low} = max(order) + 1$ and the higher $Len_{high} = max(z|z = len(z_j) : z_j \in E_z)$ by the maximum session length found at the dataset, where *len()* denotes the length of a sequence.

3.2. Markov Model

MM is defined as a stochastic process X_n having a finite number of states $n = 0, 1, 2, ...$ The process satisfied the following condition $p(i, j) = P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, ..., X_0 = i_0)$. The outgoing probabilities from a state i are defined as $\sum_{j \in S} p(i, j) = 1$ [41]. MM of *order* o defines a process $X^{(n)}$ that depends on recent o states [40]. The future steps are denoted as *distance* $d = \{1, 2, ...\}$. MM is used to provide a naïve model baseline and to compare the results of the LSTM and MLP implementations used in this study for the scale of the performance.

3.3. Deep Learning Page Predictors

The DL click event prediction is treated as a *multi-class classification* problem having the state space $|U|$ of possible outcomes. A probability function $prob_m$ is fitted during a *training* process to model the probability $p(y|x)$, where x denotes a sequence of visited pages, and y is the probability distribution of the available pages. Items on x are mapped using one-hot encoding to the vector of length m, where the mth index indicates the presence of a page $\mathbf{x} = [x_1, ..., x_m]$. The output distribution $\mathbf{y} = [y_1, ..., y_m]$ represents the probabilities of the subsequent pages having the same dimension as the input. The outcome selected as an event with maximum probability (2) [42].

$$\hat{y} = \arg \max_{m \in \{1...|U|\}} prob_m(\mathbf{x}). \tag{2}$$

Multilayer perceptron (MLP) is a network of perceptrons constituting at least an input layer, a hidden layer and an output layer. The learning goal is to find the weights w and biases b so that the output approximates the $p(y|x)$ for all training inputs. Given inputs $x = \{x_1, x_2, ..., x_m\}$, bias b, and weights $w = \{w_1, w_2, ..., w_m\}$, the output of a single layer l having a multiple inputs is defined in (3) [43]. We use concatenation for inputting a number

of past events to the model designated by o. The input x may then have multiple one-hot encoded pages depending on the number of o.

$$y_l = \sigma(\sum_{i=1}^{m} w_{li} x_i + b). \tag{3}$$

Long short-term memory (LSTM) networks are based on gated memory units. They are designed for sequential data to model the temporal dependencies of variables. The main idea is to learn whether to keep or forget previous values in time so that the memory would not vanish during the *gradient descent* [44]. LSTM network is presented in (4) to (9).

$$i_t = \sigma(W_i \cdot h_{t-1} + V_i \cdot x_t + b_i), \tag{4}$$

$$o_t = \sigma(W_o \cdot h_{t-1} + V_o \cdot x_t + b_o) \tag{5}$$

$$f_t = \sigma(W_f \cdot h_{t-1} + V_f \cdot x_t + b_f), \tag{6}$$

$$\tilde{C}_t = tanh(W_c \cdot h_{t-1} + V_c \cdot x_t + b_c), \tag{7}$$

$$C_t = i_t \odot \tilde{C}_t + f_t \odot C_{t-1}, \tag{8}$$

$$y_t = o_t \odot tanh(C_t). \tag{9}$$

C_t is the cell state and $\tilde{C}_t$ is the candidate whether the information should be persisted or not. x_t and y_t are the input and output. W are the weight matrices of the input, output, and forget gates, as well as the cell state. V is the same for inputs and b the bias vectors. i, o, and f are the input, output, and forget gate vectors [44]. It can be seen that the LSTM is clearly more complex in comparison with the MLP network. However, training the LSTM model do not require similar concatenation as MLP as it natively supports a sequence of $\mathbf{x}$ as an input.

The training for each *order* splits each sequence s_j to a training input sequences, where $|x_{train}| = order$ is the number of model input events and $|y_{train}| = 1$ is the number of model output events. Similar procedure exists for the inference, where a x_{test} is generated for the model input. Corresponding ground truth event $|y_{test}| = 1$ is taken and compared with model output $\hat{y}$. A generator pattern is used to predict the subsequent pages designated by the d, where an output of a stage is fed as an input of the second stage.

3.4. Session-Trajectory Learning

Q-learning is off-policy TD RL algorithm. Its approximation of optimal policy Q^* is independently estimated given behavioral policy. The policy update is done at time $t+1$ using observed reward R_{t+1} and the discounted value estimation. The original value of the state-action pair is then updated with an error (difference) between a look ahead value estimation and the previous estimated value (11). The update is controlled using step-size parameter α, which controls the learning rate and with discount-rate γ of the $t+1$ value estimation [7]. We found out that the TD is the most essential feature for our *session-trajectory learning* not to forget previously learned state transitions.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \tag{10}$$

Our target is to learn a directed web graph $G = (V, E)$ where each node represents a click event in our dataset and vertex a transition from event e_t to e_{t+1}. The graph models the state transition probabilities by maximizing a reward function of state-action pairs $r(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s']$ [7], where R_t is the reward and S_t is the state at time t proceeding state S_{t-1} and action A_{t-1}. We treat the reward function equal to the sequence-aware recommender utility function and introduce a novel algorithm for training its estimation, which draws inspiration from two different branches of research: RL-based graph generating solutions [25,26] and RL recommender algorithms through hierarchical clustering [33].

We name the algorithm as *session-trajectory learning*, which is a graph generation procedure using Q-learning design, where the goal is to approximate the reward function using an estimation policy $Q(S_t, A_t)$. The algorithm proceeds in discrete episodes iterating the clickstream sequences z_j in order from the first event to the last. By having user guided trajectories, i.e., the page transitions of past clickstreams towards the target leaf pages, we model the multi-step page transitions from any given start page to popular leafs in our final graph. The process is described with a set of states S, actions A, and consequential rewards $R(a_t)$ at time t, where S represents the events in clickstreams, A describes their state transitions as actions, and $R(a_t)$ is a reward function for a state transition.

We design an *environment* that fits our hierarchical page domain providing *session trajectories* in each episode, i.e., sample sequences z_j from our sequence database $z_j \in E_z$, where each z_j is selected to contain only sequences that ends with a leaf page that has no child pages. The annotations for each sample follow a rule, where each subsequent click event $e_{i<k}$ in sequence z_j is an intermediate state and the last e_k as our target state e_{tgt}, which in turn is a *terminal* event in an episode.

The *session trajectory* differs from traditional RL trajectories in a single episode [7], in a such way, that we expect the past sessions generate trajectories that are independent of each other and may end in different terminal states where the start state during the inference may be any of the intermediate states along a session trajectory, which will eventually decide the target. In an extreme case, we only see unique trajectories during the training.

State represents an event e_i in the given sequence z_j in an episode. The RL agent observes the states by iterating a clickstream state transitions from the first event to the last. Non-terminal, intermediate events are between the start and terminal state. The terminal state indicates last event of the sequence.

Action describes a single state transition two click events in a single trajectory. We generate the final graph by observing single trajectories, where each action is also a modification to the final graph where an edge is created between two nodes.

Reward design constitutes a three level of different reward values. A negative value is given whether the transition is illegal. A small positive reward is given to intermediate actions and a large reward when the terminal state is reached.

During training, represented in Algorithm 1, actions are generated for a given state using the ϵ-greedy behavioral policy, where ϵ represents the probability whether a random action is selected. Otherwise, an action indicated by the estimation policy is used. A balance between the exploration and exploitation is smoothed by using decaying ϵ for each episode.

Algorithm 1 Session trajectory learning

Input: Training data, click sequences $\{z_1, z_2, ..., z_h\}$
Initialize learning rate and discount factor $\alpha \leftarrow 0.9$, $\gamma \leftarrow 0.1$
for all z_i in z_h **do**
    Initialize $S_t \leftarrow$ first event e_0 of z_i
    for all e_i in z_i **do**
        Select A_t at S_t, the state transition e_i to $\hat{e}_{i+1}$ derived from Q using ϵ-*greedy*
        Compare $S_{t+1} = \hat{e}_{i+1}$ with ground truth e_{i+1} in z_i
        if $\hat{e}_{i+1} = e_{i+1}$ and terminal **then**
            $R_{t+1} \leftarrow 1$
        else if $\hat{e}_{i+1} = e_{i+1}$ and intermediate **then**
            $R_{t+1} \leftarrow 0.1$
        else
            $R_{t+1} \leftarrow -0.1$
        end if
        $Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$
        $S_t \leftarrow S_{t+1}$
    end for
end for

The RL agent iterates the sequences via the environment. An episode start initializes the environment to a first state of the current sequence. Stepping the environment it proceeds to the next state by using an action generated by the behavioral policy. If given state transition was correct in the current sequence, the agent receives a small positive reward for the intermediate states and large if the correct terminal is reached. If not, the environment stays in the current state with returning a negative reward.

The goal of the algorithm is to maximize the cumulative rewards so that during the inference the path from the context to the expected target page can be recalled. Due to the independence aspects of each trajectory, we found out that a single random trajectory may override a crucial path that resembles strong association between any given intermediate and target state. Therefore, we must apply a function for the estimation policy that does not override the past associations, but calculate smooth updates for the policy per each episode. To achieve this, we found that the temporal-difference (TD) is the most essential feature for the estimation policy update. Having the smoothed update present, the total cumulative reward from any context event to the terminal will then represent an ideal state transition sequence from a given context to a target event, i.e., a leaf content page.

3.5. Top-N Experimentation

The nature of the hierarchical information structure presents a multi-choice selection for the following events. If a context event e_{ctx} ($e_{ctx} \in E_{test}$, i.e., the current page of an active user) is selected, the predicted next event e_p is sampled from a *distribution* of the child pages $E_c = \{e_p | E_c \subset U\}$. In other words, the predicted event e_p, or the following events e_{p+d}, may have *multiple labels*, from which the right choice depends on a latent information need. If events with the highest probability are only assumed to be relevant, the outcome is optimistic by design because a popular choice may not be the right one. The effect needs to be verified by applying a test setup to confirm, whether an event of choice is generated by the model, but with a lower probability. An experimental top-N setting is prepared to prove, whether we can improve the accuracy of the user selection. In that case, an additional feature engineering would be applicable to improve the performance of the modeling.

Top-N experiments are run with options 1, 3, and 5 to observe whether relevant events exist below what are suggested by the maximum probability. The *multi-class classification* model provides probabilities for next events given a single event or a list of input events. Taking into an account the top-N list of probabilities, pseudo relevance feedback can be assessed by selecting and testing, whether the ground truth event exists in the top-N probable events.

If the model is successful, it learns a variety of click patterns and predicts actual users' behavior correctly. This would suggest that the model could be further enhanced by additional features. A selection-based algorithm is used at the inference for the top-N experiment. It uses a ground truth event selected from the test set. If it is in the predicted top-N events, the event is returned. If the top-N list do not include the ground truth event, the default event with the highest probability is selected.

4. Evaluation

4.1. Data

The data in this study consist of the web pages and the session logs of the public website from the Finnish city of *Tampere*. The logs are collected during two full years, 2011–2012. Each year contains 6.9 million sessions per year. The state space, i.e., the number of unique URLs is 245,532, containing referring pages from domains outside the site, i.e., from social network applications, newspaper web sites, etc. URLs are also dynamically generated by the website search.

The portal contains 225,000 web pages, including 105 different topics under the root URL domain, i.e., counted as the first child nodes of the root URL. However, the topics include internal, duplicate, under construction, etc., type of information that is not related

to common usage. Commonly used topics include: *civil engineering*, *philharmonic orchestra*, *education*, *culture and museums*, *health services*, etc. The page visits are not evenly distributed across the year. Figure 1 illustrate three typical visit patterns: low and high holiday season usage and even distribution which may include biased months.

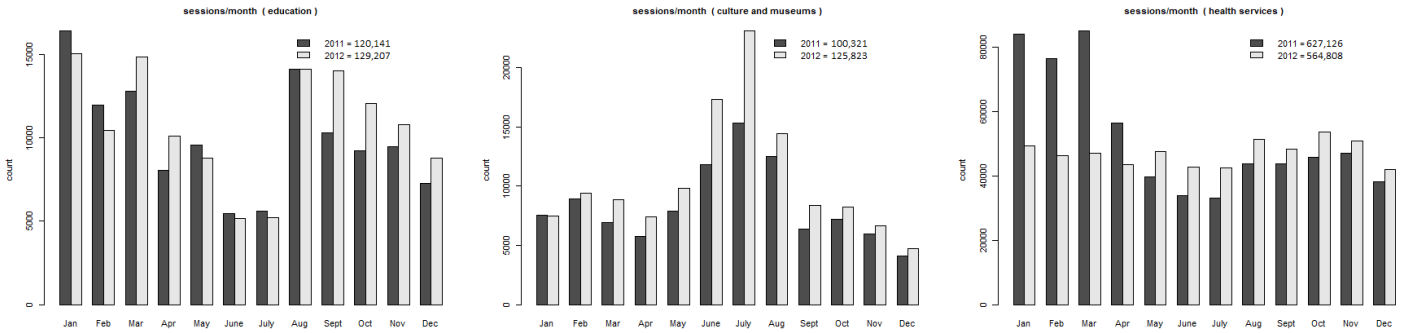

Figure 1. Typical visit patterns.

There are time dependencies on the consumption of popular topics, as well as single pages. The statistics of only the most popular topics will give an overview of the dependencies. Figure 2 presents the top-5 most popular topics per month. *Sports and leisure* (orange), and a local museum *Vapriikki* (blue), as well as *health services* (grey), illustrates the constant consumption through the year. *Culture and museums* (brown) and *education and studies* (light blue) appear only a limited amount of time in top-5 list.

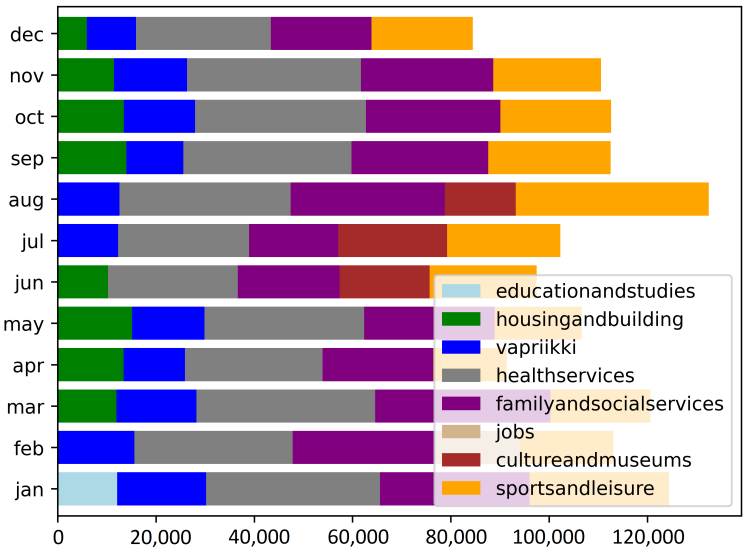

Figure 2. Annual monthly top-5 topic consumption.

The *Bouncing rate* (a ratio of single and total number of page visits) is high as 85%, i.e., the large portion of the data contains a single page sessions. The distribution of session lengths ranges from a single click up to one hundred in a yearly basis with average of 35 clicks, the mode of the session lengths is between 1 and 15 clicks, where the single page visits dominate.

The hierarchy of the topics structure is represented in the slash separated path segments of the URLs: *root/education/primaryeducation/curriculum.html*. There are 81 different topics (unique segments as a first child) under the root domain, including non-public system related content, as well as redundant topics that are temporary for page updates. Sessions, including visits in 26 unique topics, are selected for the study. Counting the path lengths from URLs used in the sessions, the pages form a hierarchy of ten levels deep. Distribution of visited URL depths is presented in Table 3.

Table 3. URL depths.

Depth	1	2	3	4	5	6	7	8	9	10
count	62	153	931	12,507	4205	7528	677	126	19	5

4.2. Measures

Evaluation focuses on the prediction accuracy of the subsequent pages given a random context page or a history of o pages. The data are divided using the common 80/20 random split for training and testing, respectively.

The training of the models is done by sampling from the population of the year 2011. This is done to gain an overall understanding about the applicability to support a generic model for the data and the overall performance of the modeling. *Orders (o)* 1 to 3 are used to test the performance of the model when adding more memory, i.e., the effect of previous clicks. *Distances (d)* from 1 to 6 are used for each model that indicates capability to predict a number of steps forward to define a window of future predictions based on the URL depth statistics. The majority of the pages are at four to six deep hierarchies.

During testing, a next event is predicted as described at the modeling section. In the evaluation of the performance, the predicted event is compared with the ground truth event. A random page e_c is selected from each test set session, which is treated as the current *context page* of an active user. The ground truth page e is selected from the same session and a number of steps are counted forward from e_c indicated by the parameter d. The trained model is used to predict the next page $\hat{e}$ from the given e_c for each d. As described in the modeling, we run top-N experiments which are constructed to test whether there is an event at top-N probable events corresponding the ground truth.

A list of $< e, \hat{e} >$ pairs is constructed and used as an input to statistical evaluation having the number of examples of the whole test set. The predictions are treated as the *multi-class classification* problem, where each web-page represents its own class.

Accuracy is chosen as the main evaluation criteria with the addition of and *F1* to indicate the overall performance taking the *precision* and *recall* into an account. The approach to evaluate overall multi-class performance includes the calculation of *macro/micro averaged value* across all classes. The *macro* approach is an average of each metric over all classes. *Micro*, on the other hand, includes the aggregation of all contributing values and then calculating the metrics. Like Refs. [45,46], the evaluation in this paper is done using the *macro averaged value*.

4.3. Evaluation Setup

Sequence lengths from minimum Len_{low} and maximum of Len_{high} are used for training. The Len_{low} requirement is due to the *order (o)* of the model. The model of o requires $o + 1$ of minimum the sequence length, i.e., the number of click events that corresponds o and the number of events that corresponds the number of *distance (d)* of forward clicks.

Three main rules for the data selection are applied: (a) sequences starting on the root page and traversing the natural topic hierarchy of the structure, (b) random selection of sequences of 26 selected topics from the population of the year 2011, and (c) sequences ending on a leaf page of the page hierarchy. We choose the latter rule as an assumed relevance assessment that user arriving at a leaf page has found the information of interest.

It was found out that the dataset contains many internal pages to the web hosting service, in addition to topic pages that are used in publishing new content that are duplicates of some topics, but are consumed in different time during the year. From the total number of topics, there are ten the most popular top-5 that appear constantly during the year. Additionally, there are short-lived pages that appear a limited time to promote an event type of content.

We wish to concentrate on the most relevant content pages that may appear on the most popular pages, but raise up beneficial pages that may appear on a larger number of topic selection. A subset of 26 topics are then chosen because internal pages and otherwise

redundant page information exists at the session data. Technical reasons also play a role for the filtering. The number of sessions, as well as the large state space pose large computing requirements for the modeling.

We evaluate the ability of each of the model to predict the future page transitions towards the assumed target pages at the leaf nodes of the page hierarchy. The depth of the page topology was found to be ten pages deep having most of the pages at four to six deep hierarchies. By knowing the depth distribution, the distance from one to ten is used in the experiments to illustrate the model prediction performance down the hierarchy.

The MM model operations are based on an *R language* implementation of *clickstream* package [40] executed on *Intel i7 6700K* CPU architecture. DL modeling is a *Python*-based implementation using the *Tensorflow* library utilizing *Nvidia GTX 1080 Ti* GPU. MLP designs are compared with LSTM architectures. Table 4 lists the used DL model parameters. The effect of number of layers and neurons to the accuracy are used for the evaluation.

Table 4. DL model parameters.

Parameter	Value
Input Layer Neurons	The state space
Hidden Layers	1 to 3
Hidden Layer Neurons	32, 256 and 1024
Output Layer Neurons	The state space
Activation Function	Softmax
Loss Function	Categorical Crossentropy
Learning Rate	0.001 to 0.00003
Weight Initialization	Xavier
Epochs	300

5. Results

This section presents the effect of the order for our dataset and the performance comparison between the MM, LSTM, MLP, and RL models having dataset as an input containing random samples of the population of the year 2011. In each of the figures, *accuracy* and *F1*-scores are shown in percentages in relation to the distance prediction for each of the models.

The experimentations are run by subsetting the full dataset of magnitude of $|S| = 14M$ and $|U|$ = 245,000, from which we select 26 topics to be able to focus on content and additionally reduce the data for computing efficiency reasons. Then, we choose the states from $o = 1$ to $o = 3$ as the memory of the models. We also use the session length of $Len_{low} = 4$ to $Len_{high} = 100$ to ensure that the single page sessions are not used in evaluation. Total amounts reduces then to $|S_{26}|$ = 22,113 and $|U_{26}|$ = 11,193.

Figures 3 and 4 presents the results of the first experimentations, where we investigate the effect of the order for our dataset by using MM and DL baseline classifiers. We can see the lower than ten percent improvement or none in accuracy for the immediate, low order predictions. The overall prediction performance drops significantly for further distances being close to zero at $d = 4$.

In the second experiment, we present the performance comparison between the baseline and our RL-based models by using $o = 1$ models. In Figures 5 and 6, we can first see the same effect of dropping accuracy, as well as overall performance, for each of three baseline methods for further distances. Additionally, increasing the size of the DL layers does not improve their accuracy. The RL method performance does not improve the initial accuracy; however, it shows it is rather stable across the distances improving the multi-step prediction capability.

Figure 3. Accuracy related to the order of baseline models.

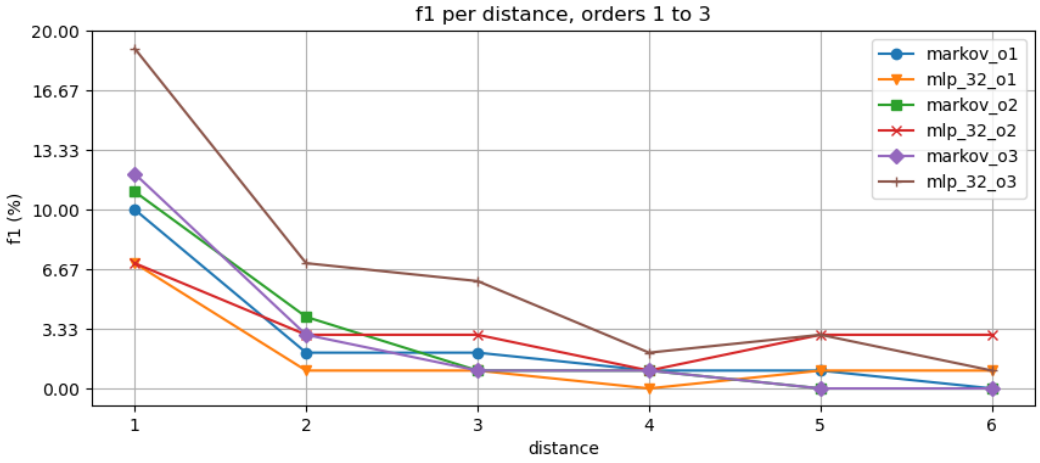

Figure 4. F1 related to the order of baseline models.

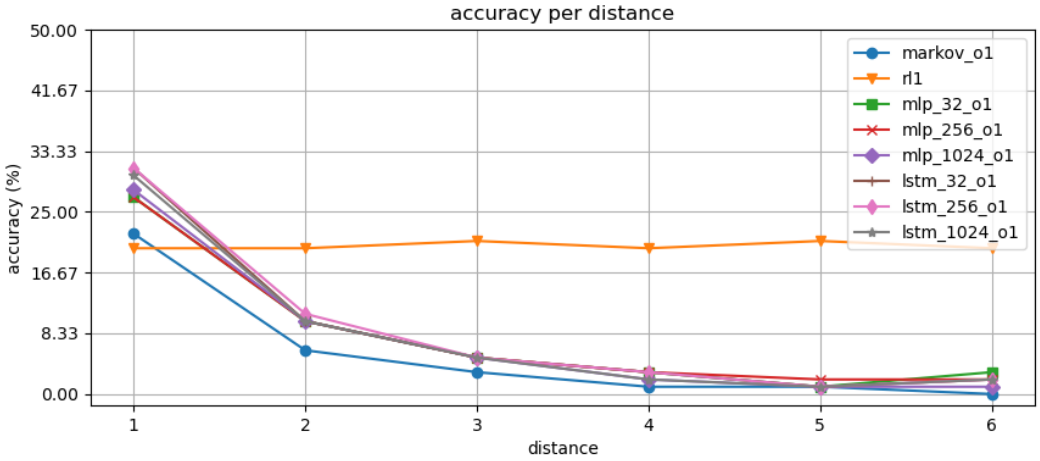

Figure 5. Accuracy comparison between the baseline and the RL-based models.

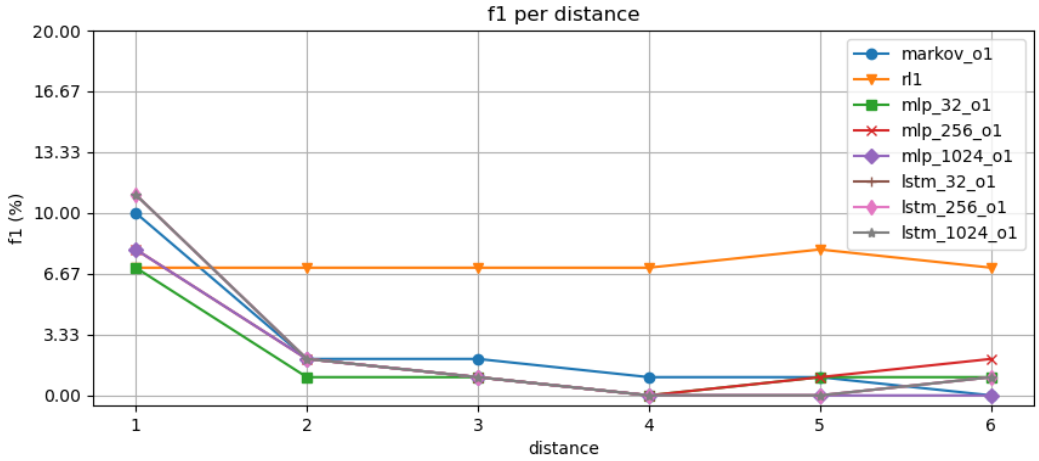

Figure 6. F1 comparison between the baseline and the RL-based models.

Finally, we observe the experimental result of top-N lists using a simple MLP model, whether there are matches below the suggested maximum probability. In Figures 7 and 8, we see an initial increase in the number of first-step matches—roughly a 10% increase between top 1, 3, and 5 lists. This still shows the effect of the $d > 1$ predictions with dropping accuracy under an average range of 10%.

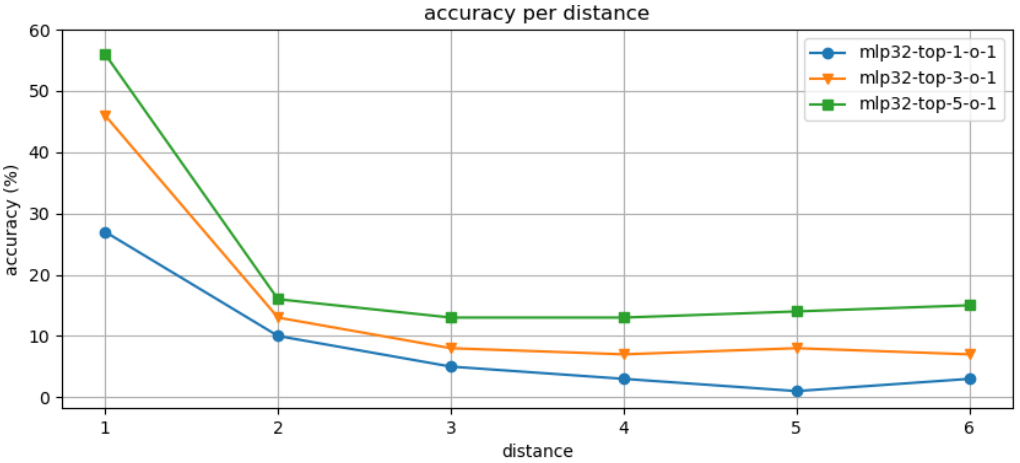

Figure 7. Accuracy of top-N experimentation.

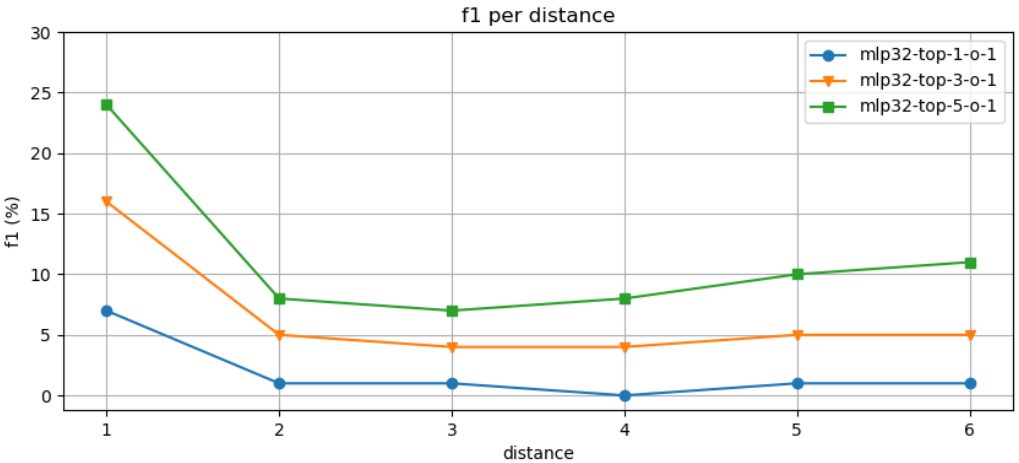

Figure 8. F1 of top-N experimentation.

6. Result Analysis

We started the evaluation by assessing a baseline performance for our dataset using Markov Model, LSTM, and MLP classifiers. In the first experiment, we see in Figures 3 and 4 similar performance for each of the classifiers, i.e., the DL-based classifier performance is same order of magnitude for each distance. The accuracy in Figure 3 varies between 7% and 10% for $o = 1$, 7% and 11% for $o = 2$, and 10% and 12% for $o = 3$ for each of the models. The models overall F1 performance Figure 4 range is between 7% and 19%. The order (memory) of the model did not increase the accuracy considerably. We also see that the distance (prediction horizon) usually decreases the accuracy 10–20% per step. We can only consider MM as a stable baseline for its growing state space complexity issue [6].

The second experimentation, we explore the size of the DL classifiers and compare the classifiers' performance against our RL-based implementation. For the first experiment, the number of neurons are varied (2^5, 2^8, and 2^{10}). Approximately 22–30% accuracy can be achieved by the sequential model using only the pages with the highest probability. The performance can be seen in the Figures 5 and 6, where the same accuracy 10–20% per step decrease as in previous experiment, i.e., the size of the DL models will not increase the performance. It will relate to a rather poor user experience if used as such. However, with our RL-based implementation, we can see a stable 20% accuracy for each distance. We would have the better ability to predict the target leaf pages at the lower levels of the page hierarchy.

We concluded the evaluation using an experimental setting to validate, whether the samples of top-N probabilities include the correct matches in comparison with the ground truth. Figures 7 and 8 illustrate that there are samples of correct matches as we investigate the lower probability range, even if we see the similar steep drop in accuracy for future predictions after the first-step matches.

We saw at the start of the experiments, that the *order* (memory) of the model did not increase the accuracy considerably. This may be explained that the relevance of the next click is already a multi-choice, and the history is not an apparent hint of the future. The same effect seems to apply for the future steps. The future prediction accuracy seems to be stable in 20–30% range for immediate steps $d = 1$ but low for further $d > 1$ steps for baseline classifiers. The RL-based algorithm seems to have more stable performance $d > 1$ steps. This would ensure that we can achieve stable leaf node predictions only using just the sequence information as a feature. The potential additional feature engineering was suggested by the top-N experimentations.

7. Conclusions

In this paper, we assessed the performance of Markov, deep learning and Q-learning models in order to predict click events based on the clickstreams of a hierarchical website. We evaluated the accuracy of the classifiers by observing a random page selection and observed the effect of the history and the prediction horizon for future click events. It was found out that 22–30% accuracy can be achieved by using the baseline classifiers for the set of data. The history of a session did not have any considerable effect on the modeling performance and the prediction horizon for subsequent steps of two to three was approximately 2% better in comparison with the single step accuracy. Our RL-based algorithm, however, has more stable performance for the future steps.

The modeling problem for the hierarchical website is a *multi-class* problem, having a state space of pages down the hierarchy. The clickstreams contain information about the user traversal patterns. They contain the paths to the relevant pages, but they are open to interpretations as each following click events has a number of outcomes. Not only the problem has the multi-class property, but also a *multi-label*, where each subsequent event has multiple labels attached depending on the latent information need.

Popularity bias, as defined, for example, in References [8–10], weights the pages that are often visited. The secondary events in a rank of probabilities contain relevant information about the subsequent pages that eventually lead to the needed content. This

was observed in top-N experiments. Not only the past visited pages hint the probability of the next, but they are a source of bias. The information need is also a latent feature, i.e., not observable directly in the sequential domain. The history is neither a univocal facet describing the information need.

However, it was found out that in our domain we have usually four to six and up to ten level deep page hierarchies, which contains the target, content pages at the leaves of the hierarchy. It is not feasible to recommend intermediate pages which are used to navigate to content pages. Therefore, there is a need to provide accurate recommendations to multiple levels forward. Q-learning proved a stable performance for multi-step distances compared with the baseline models. Further, it shows a promising result for our solution for the specific problem domain to continue our work.

We have considered the hierarchical page topology in our design for a particular domain scenario, but as described in the modeling chapter, our goal was to generalize the use in training for wider use as long as the topology is presented in the training data. Typical browsing behaviors may include a chain type in hierarchical structures and star for social network type of topologies and mixed for more complex scenarios [38], which may be used as a guideline to test the applicability of our algorithm in wider scope. In order to investigate the fit for general use, a dedicated study may be appropriate for the proof using a wider set of common datasets.

In the future, we would like to explore the feature space of the user sessions to improve the multi-step performance of Q-learning approach. We would consider of exploring the session features to maintain the user anonymity. The features, such as time, the label of a topic, content, etc., may all be candidates to enhance the prediction accuracy of multiple subsequent clicks on the hierarchical web domain. A set of proven session features would then be used for accurate page recommendations for an anonymous user.

Author Contributions: Conceptualization, P.P.; methodology, P.P.; software, P.P.; validation, P.P.; formal analysis, P.P.; investigation, P.P.; resources, P.P.; data curation, P.P.; writing—original draft preparation, P.P., K.S., J.K., M.J.; writing—review and editing, P.P., K.S., J.K., M.J.; visualization, P.P.; supervision, K.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Quadrana, M.; Cremonesi, P.; Jannach, D. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* **2018**, *51*, 1–36. [CrossRef]
2. Ntoutsi, E.; Stefanidis, K. Recommendations beyond the ratings matrix. In Proceedings of the Workshop on Data-Driven Innovation on the Web (DDI '16), Hannover, Germany, 22–25 May 2016; Association for Computing Machinery: New York, NY, USA; pp. 1–5. [CrossRef]
3. Su, X.; Khoshgoftaar, T.M. A Survey of Collaborative Filtering Techniques. *Adv. Artif. Intell.* **2009**, *2009*. [CrossRef]
4. Shi, Y.; Larson, M.; Hanjalic, A. Collaborative Filtering beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Comput. Surv.* **2014**, *47*, 1–45. [CrossRef]
5. Wu, Z.; Tan, B.H.; Duan, R.; Liu, Y.; Goh, R.S.M. Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns. In Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge), Vienna, Austria, 16–20 September 2015. [CrossRef]
6. Eirinaki, M.; Vazirgiannis, M.; Kapogiannis, D. Web path recommendations based on page ranking and Markov models. In Proceedings of the 7th ACM International Workshop on Web Information and Data Management (WIDM 2005), Bremen, Germany, 5 November 2005; Association for Computing Machinery: New York, NY, USA; pp. 2–9. [CrossRef]
7. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; A Bradford Book; MIT Press: Cambridge, MA, USA, 2018.
8. Borges, R.; Stefanidis, K. Enhancing Long Term Fairness in Recommendations with Variational Autoencoders. In Proceedings of the MEDES 2019: The 11th International ACM Conference on Management of Digital EcoSystems, Limassol, Cyprus, 12–14 November 2019; pp. 95–102
9. Borges, R.; Stefanidis, K. On Measuring Popularity Bias in Collaborative Filtering Data. In Proceedings of the BigVis 2020: 3rd International Workshop on Big Data Visual Exploration and Analytics, Copenhagen, Denmark, 30 March 2020.

10. Borges, R.; Stefanidis, K. On Mitigating Popularity Bias in Recommendations via Variational Autoencoders. In Proceedings of the 36th ACM/SIGAPP Symposium On Applied Computing, Gwangju, Korea, 22–26 March 2021; pp. 1383–1389
11. Jindal, H.; Sardana, N. Web navigation prediction based on dynamic threshold heuristics. Available online: https://www.sciencedirect.com/science/article/pii/S1319157820303244 (accessed on 28 May 2021). [CrossRef]
12. Fedushko, S.; Ustyianovych, T.; Syerov, Y.; Peracek, T. User-Engagement Score and SLIs/SLOs/SLAs MeasurementsCorrelation of E-Business Projects Through BigDataAnalysis. *Appl. Sci.* **2020**, *10*, 9112. [CrossRef]
13. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System: A survey and new perspectives. *ACM Comput. Surv.* **2019**, *52*, 1–38. [CrossRef]
14. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web (WWW '17), Perth, Australia, 3–7 April 2017; pp. 173–182. [CrossRef]
15. Gharibshah, Z.; Zhu, X.; Hainline, A.; Conway, M. *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*; Springer: Berlin/Heidelberg, Germany; pp.196–204.
16. Guo, C.; Berkhahn, F. Entity embeddings of categorical variables. *arXiv* **2016**, arXiv:1604.06737.
17. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), Melbourne, Australia, 19–25 August 2017; pp. 1725–1731.
18. Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; Chua, T.-S. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), Melbourne, Australia, 19–25 August 2017; pp. 3119–3125.
19. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M. et al. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systemsm Boston, MA, USA, 15 September 2016; pp. 7–10. [CrossRef]
20. Waradpande, V.; Kudenko, D.; Khosla, M. Deep Reinforcement Learning with Graph-based State Representations. *arXiv* **2020**, arXiv:2004.13965.
21. Perozzi, B.; Al-Rfou, R.; Skiena, S. DeepWalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), New York, NY, USA, 24–27 August 2014; pp. 701–710. [CrossRef]
22. Grover, A.; Leskovec, J. Node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 855–864. [CrossRef]
23. Zhou, C.; Liu, Y.; Liu, X.; Liu, Z.; Gao, J. Scalable graph embedding for asymmetric proximity. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), San Francisco, CA, USA, 4–9 February 2017; pp.2942–2948
24. D'Arcy, L.; Corcoran, P.; Preece, A. Deep Q-Learning for Directed Acyclic Graph Generation. *arXiv* **2019**, arXiv:abs/1906.02280.
25. You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montreal, QC, Canada, 3–8 December 2018; pp. 6412–6422.
26. Wan, G.; Pan, S.; Gong, C.; Zhou, C.; Haffari, G. Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In Proceedings of the 29th International Joint Conference on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; pp. 1926–1932.
27. Taghipour, N.; Kardan, A.; Ghidary, S.S. Usage-based web recommendations: A reinforcement learning approach. In Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys '07), Minneapolis, MN, USA, 19–20 October 2007; pp. 113–120. [CrossRef]
28. Cai, Q.; Filos-Ratsikas, A.; Tang, P.; Zhang, Y. Reinforcement Mechanism Design for e-commerce. In Proceedings of the 2018 World Wide Web Conference. *arXiv* **2018**, arXiv:1708.07607.
29. Zou, L.; Xia, L.; Du, P.; Zhang, Z.; Bai, T.; Liu, W.; Nie, J.-Y.; Yin, D. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20), Houston, TX, USA, 3–7 February 2020; pp. 816–824. [CrossRef]
30. Wang, X.; Xu, Y.; He, X.; Cao, Y.; Wang, M.; Chua, T.-S. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In Proceedings of the Web Conference 2020 (WWW '20), Taipei, Taiwan, 20–24 April 2020; pp. 99–109. [CrossRef]
31. Lei, Y.; Pei, H.; Yan, H.; Li, W. Reinforcement Learning based Recommendation with Graph Convolutional Q-network. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), Xi'an, China, 25–30 July 2020; pp. 1757–1760. [CrossRef]
32. Zhang, J.; Hao, B.; Chen, B.; Li, C.; Chen, H.; Sun, J. Hierarchical Reinforcement Learning for Course Recommendation in MOOCs. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), Honolulu, HI, USA, 27 January–1 February 2019; pp. 435–442. [CrossRef]
33. Chen, H.; Dai, X.; Cai, H.; Zhang, W.; Wang, X.; Tang, R.; Zhang, Y.; Yu, Y. Large-scale Interactive Recommendation with Tree-structured Policy Gradient. *arXiv* **2018**, arXiv:abs/1811.05869.
34. Nadeau, D.; Sekine, S. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investig.* **2007**, *30*, 3–26. [CrossRef]

35. Otter, D.W.; Medina, J.R.; Kalita, J.K. A Survey of the Usages of Deep Learning in Natural Language Processing. *arXiv* **2018**, arXiv:1807.10854.
36. Stratigi, M.; Nummenmaa, J.; Pitoura, E.; Stefanidis, K. Fair Sequential Group Recommendations. In Proceedings of the SAC'20: The 35th ACM/SIGAPP Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 1443–1452
37. Bouraga, S.; Jureta, I.J. Knowledge-based recommendation systems: A survey. *Int. J. Intell. Inf. Technol.* **2014**, *10*, 1–19. [CrossRef]
38. Ramaciotti Morales, P.; Tabourier, L.; Ung, S.; Prieur, C. Role of the Website Structure in the Diversity of Browsing Behaviors. In Proceedings of the 30th ACM Conference on Hypertext and Social Media, Bavaria, Germany, 17–20 September 2019; pp. 133–142. [CrossRef]
39. Zheng, G.; Peltsverger, S. Web Analytics Overview. In *Encyclopedia of Information Science and Technology*, 3rd ed.; IGI Global: Hershey, PA, USA, 2015.
40. Scholz, M. R Package clickstream: Analyzing Clickstream Data with Markov Chains. *J. Stat. Softw.* **2016**, *74*, 1–17. [CrossRef]
41. Jiang, J. Stochastic Processes. In *Large Sample Techniques for Statistics*; Springer Texts in Statistics; Springer: New York, NY, USA, 2010.
42. Sugiyama, M. *Introduction to Statistical Machine Learning*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2015.
43. Parr, T.; Howard, J. The Matrix Calculus You Need For Deep Learning. *arXiv* **2018**, arXiv:1802.01528.
44. Petneházi, G. Recurrent Neural Networks for Time Series Forecasting. *arXiv* **2018**, arXiv:1901.00069.
45. Tsoumakas, G.; Katakis, I.; Vlahavas, I. Mining Multi-label Data. *Data Mining and Knowledge Discovery Handbook*; Springer: Boston, MA, USA, 2010. [CrossRef]
46. Zhang, M.L.; Zhou, Z.H. A Review on Multi-Label Learning Algorithms. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1819–1837. [CrossRef]