UNIVERSITY OF AMSTERDAM

MSc ARTIFICIAL INTELLIGENCE
MASTER THESIS

A Non-Sequential Neural Click Model for Web Search

by
MARTIJN WARDENAAR
10348190

December 29, 2017

36 EC
January 2017 - January 2018

Supervisor:
Dr. I. MARKOV

Assessor:
Dr. I. MARKOV
Prof. Dr . M. DE RIJKE

UNIVERSITY OF AMSTERDAM

Abstract

Getting a better understanding of click behavior is important for advancing information retrieval systems. A lot of work has already been done on click models that predict what documents a user will click on, but these models do not take order into account. The aim of this work is to create a click model that not only predicts what documents of a Search Engine Result Page (SERP) the user will click on, but also in what order. To do this two Deep Learning models are implemented and compared to the recently presented Neural Click Model. The Neural Click Model (Borisov et al., 2016) was recently introduced and is based on neural networks and managed to outperform existing click models that rely on the Probabilistic Graphical Model (PGM) framework. Both new models are adaptations of the Neural Click Model and use a bidirectional Long Short-Term Memory (LSTM) module to encode the information about the query and the documents. One of the models also uses an attention mechanism to give more information about the documents at the time of predicting the clicks. The results show that predicting the order of clicks in a sequence is a very complex task and that overall it is difficult to improve on models that assume a linear examination and click path. Still it is shown that it is possible to create a neural model that is able to produce non sequential click sequences and performs on par with models that assume a sequential click sequence. This makes the non sequential neural click model a promising subject for further research. The second contribution of this research is to introduce a method for evaluating the performance of click models on non sequential click sequences. Since frequently used metrics as the likelihood and perplexity offer an unfair advantage to models with outputs that have no information about the order, two new metrics are introduced: The Damerau-Levenshtein Distance and the Longest Common Subsequence. Both these metrics compare two ordered lists and calculate a score that represents how close one is to the other while keeping order into account.

Acknowledgements

Firstly, I would like to thank my daily supervisor Ilya Markov. Throughout the thesis, which took quite longer than anticipated, he continued to make time for me to give me advise and feedback. Even when he was not in the country he was there to answer my questions and go over my progress.

A special thanks goes out to Alexey Borisov. Even though he was not officially my supervisor, he made time to help me out with this project. Apart from giving advice and brainstorming with Ilya and myself he provided the data without which this project would not have been possible.

I would like to thank prof. Maarten de Rijke for taking the time to read my thesis and be a part of the defense committee on short notice.

I would also like to thank my fellow students with whom I have spent countless hours working on projects and assignments. Especially Sébastien and Joost, working with you has been a pleasure and has helped make my time at the University fruitful and enjoyable.

Lastly, I would like to thank those closest to me. My parents who have always supported me and giving me advice throughout my life. My girlfriend who has always encouraged me and has put up with my complaining when nothing seemed to work the way I intended it to. Also my sister and my closest friends, mostly for mocking me when I complain to much.

Thank you all for helping me get to where I am today.

Contents

Chapter 1

Introduction

Internet search engines play an important role in the every day use of the internet. The amount of web pages, documents and other forms of content has grown immensely over the past years and is growing at a higher rate still. Search engines help people to satisfy their information needs by indexing the web and providing users with documents that, according to their algorithm, are best suited for the query that the user entered.

One way to improve on these search engine algorithms is by getting a better understanding of the behavior of users. Click models are algorithms that try to learn which documents a user will click based on a set of features. These models can be used to rank the documents (Chapelle and Zhang, 2009) in order to enhance the probability of having the document that best satisfies the user information need on the first rank of the Search Enginge Result Page (SERP). Other applications include simulation (Xing et al., 2013; Malkevich et al., 2017) and evaluation (Chuklin et al., 2013; Markov et al., 2014).

Many click models have already been proposed (Chuklin et al., 2015, chap. 3), some of which are discussed in section 3. Most of the existing click models are based on the Probabilistic Graphical Model (PGM) framework. Recently Borisov et al. (2016) proposed a click model that uses Deep Learning (DL) to learn user behavior. This Distributed Representation (DR) approach uses vector states to capture the information need of the user and the user behavior. The DL approach outperformed existing PGM based models, therefore it is taken as the basis upon which this research is build. The improvement that is aimed for is to allow a click model, based on Deep Learning, to predict ordered click sequences. This means that the model does not assume a linear top-to-bottom scanpath, but is able to predict in which order documents are clicked.

Lorigo et al. (2006) conducted an eye-tracking study where they found that that only 34% of the scanpaths on a SERP are linear. Wang et al. (2015) found that in two large real-world datasets, the Sogou dataset and the Yandex dataset, 27.9% and 30.4% of the multi click query sessions contain non-sequential clicking behaviors. They also found that users tend not to switch directions between clicks and that they may skip a few results in the SERP. This leads to the Partially Sequential Click Model (PSCM). This model takes skips and changes in examination direction into account, however, it does not produce ordered click sequences.

During this thesis an attempt is made to create a click model based on a neural network that is able to not only predict which documents will be clicked but also in what order. The neural click model by Borisov et al. (2016) will be used to compare to using some conventional metrics as well as new metrics that better capture the order of the predictions.

1.1 Research Questions and Contributions

The purpose of this study is two-fold; an attempt is made to create a model that is capable of predicting the order in which documents are clicked and evaluation metrics are created in order to measure the performance of ordered click sequence prediction. One disadvantage of the models that assume a linear, top to bottom scanpath is that they predict on which documents a user will click but they are not capable of predicting the order of these clicks. In particular, an attempt is made to answer the following research questions.

Q_1: *Can a Deep Learning algorithm learn to predict an ordered click sequence?*

In this case the algorithm can be said to have learned to predict ordered click sequences when it is able to predict these ordered click sequences better than models that assume documents are examined linearly from top to bottom. To answer Q_1 a deep learning model is implemented that aims at predicting ordered click sequence. Such a model will be referred to as a non sequential click model, since it predicts clicks without assuming that the order is sequential.

Q_2: *Could a non sequential neural click model benefit from having an attention mechanism?*

To answer this question another model is implemented which uses an attention mechanism to better predict click sequences. These models are compared to each other and an existing neural click model by measuring the often used likelihood and perplexity metrics. However, these evaluation metrics are not capable of capturing the performance of model based on the PGM and deep learning frameworks on ordered sequences. Therefore two new metrics are implemented and used to evaluate the models. These metrics are based on existing string comparison algorithms (Damerau-Levenshtein distance and the Longest Common Subsequence) and are able to take prediction accuracy as well as order into account.

The contributions of this research are the following:

- A Deep Learning algorithm that predicts ordered click sequences.
- An implementation and evaluation of an attention mechanism for neural click models.
- Providing a means of measuring the performance of models that predict ordered sequences.

1.2 Outline

Section 2 introduces some key concepts such as the PGM framework and Recurrent Neural Networks. In section 3 some of the common click models are discussed as well as the neural click model that was introduced recently and frequently used evaluation metrics. Section 4 introduces the new approach that aims at predicting ordered click sequences. Section 5 explains the proposed metrics that are specifically designed to compare performance in predicting ordered click sequences.

Section 6 shows the results that are obtained by the models on the metrics that are discussed in section 5. Section 6 discusses the results and section 7.1 answers the research questions brought up in section 1.1. Section 7.2 touches on some of the difficulties and future prospects of the proposed models and metrics.

Chapter 2

Preliminaries

This section discusses some key concepts that are important for understanding the research presented in this thesis. A brief overview of the Probabilistic Graphical Models (PGM) framework is given, as well as an introduction into Long Short-Term Memory (LSTM) and the Adam optimizer.

2.1 Probabilistic Graphical Models

Probabilistic graphical models are a framework that allows for the creation and execution of complex models. These models consist of nodes that have dependencies and probabilities. These models can be depicted using a graph representation such as Figure 2.1.

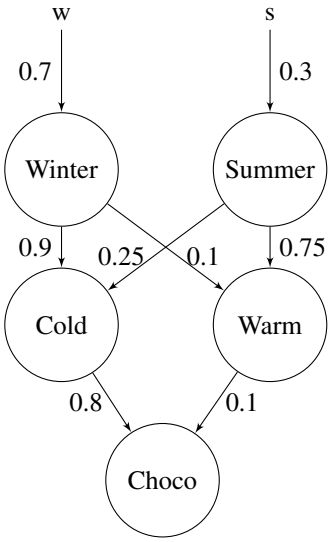


Figure 2.1: Probabilistic graphical model

A probabilistic graphical model is used to reason about the probabilies of events happening that are dependent of other events. The example given above lets us, for instance, calculate the chance of wanting an hot choco given that it is winter or the chance that it is cold given that it is summer.

$$
\begin{aligned}
P(Choco|Winter) =& P(Choco|Cold) * P(Cold|Winter) \\
&+ P(Choco|Warm) * P(Warm|Winter) \\
=& 0.8 * 0.9 + 0.1 * 0.1 \\
=& 0.73
\end{aligned}
\tag{2.1}
$$

2.2 Long Short-Term Memory

Predicting click sequences is a task where previous events may influence future events. It may be the case, for instance, that clicking on a certain document will be more likely if a previous document has been cicked but not found to be satisfying the information need. For this reason, a recurrent neural network (RNN) is more suited for the task than a regular feedforward neural network. A recurrent neural network is a network where the outcome of the previous layer influences the calculation in the current layer.

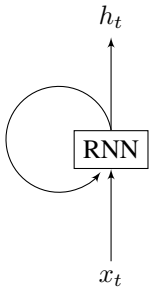


Figure 2.2: Recurrent Neural Network Model

Figure 2.2 shows a RNN. The model has three matrices, W_x, W_h and b. The output is generated using these matrices, the input and the previous state, as well as a non linear activation function (σ) such as the tanh or sigmoid. Equation 2.2 shows how a hidden state is calculated.

$$
h_t = \sigma_h(W_x \cdot x_t + W_h \cdot h_{t-1} + b) \tag{2.2}
$$

RNNs generally work well for short-range dependencies, however, for long-range dependencies they are less suited due to the *vanishing gradient problem*. The vanishing gradient problem arises when backpropagating the error of a RNN using the backpropagation through time (BPTT) algorithm. Bengio et al. (1994) showed how the training of RNNs using gradient descent is problematic for long-range dependencies due to the vanishing or exploding gradients.

Long-Short Term Memory (Hochreiter and Schmidhuber, 1997) is a model that solves the gradient problems of the RNN and allows for learning long-range dependcies. LSTMs have become very popular over the past years and are used for many tasks such as handwriting recognition (Graves et al., 2009), video recognition (Donahue et al., 2014), machine translation (Luong et al., 2014) and more. The idea behind LSTMs is a cell which stores information about past events which can be altered using a series of gates. There are a total of four gates in the LSTM: the input gate, the output gate, the input modulation gate and the forget gate.

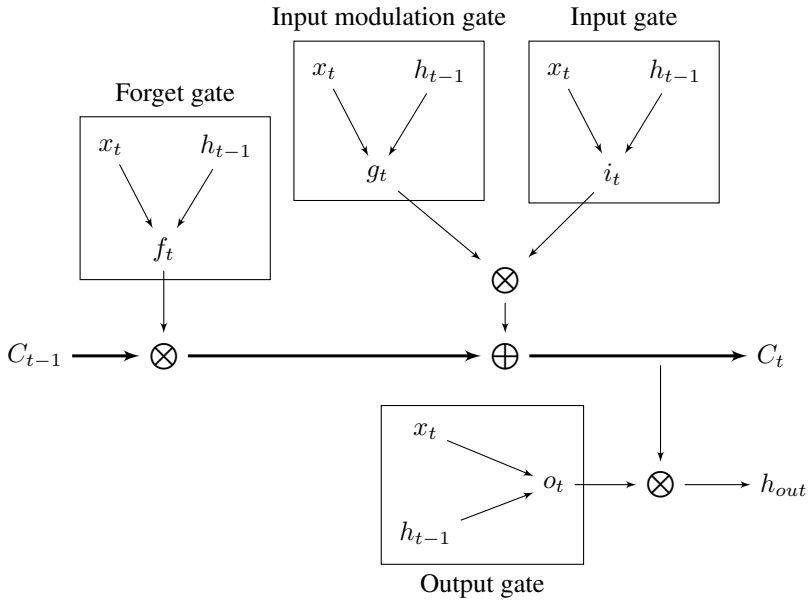


Figure 2.3: Long-Short Term Memory

In Figure 2.3 the cell state is depicted as a sort of conveyor belt where information is removed by the forget gate and added to or changed by the input gate and input modulation gate.This model is based on (Zaremba et al., 2014) which is implemented in the TensorFlow library. Equation 2.3 shows how the forget gate manages which information should be retained and which information can be forgotten.

$$C'_t = C_{t-1} \cdot \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

The adding and alteration of information is done by the input gate and the input modulation gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_t) \tag{2.5}$$

Equation 2.4 is used to decide which values of the cell state will be updated using the new information. Equation 2.5 is used to create the values that are used to update the cell state. The cell state C_t is then created using Equation 2.6.

$$C_t = C'_t + i_t \cdot g_t \tag{2.6}$$

To calculate the hidden state at time t (h_t), which is an output of the model at time t, the output gate is used. The hidden state is calculated out of two parts: the new cell state and a single layer that takes the previous

hidden state and the current input. Equation 2.7 shows the hidden layer with a sigmoid activation function that takes the previous hidden state and the current input.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.7}$$

Equation 2.8 shows how the hidden layer is computed using information in the cell state and the new information that was computed using Equation 2.7.

$$h_t = o_t \cdot tanh(C_t) \tag{2.8}$$

2.3 Adam

Neural networks need to have a method for updating network weights when training. A simple algorithm is the stochastic gradient descent algorithm, but more advanced options are available. Borisov et al. (2016) use the Adadelta optimization algorithm. During this research the Adaptive Motion Estimation (Adam) optimization algorithm (Kingma and Ba, 2014) is used. One of the advantages of Adam as well as Adadelta over the standard stochastic gradient descent is that the learning rate is adaptive.

Adam computes an adaptive learning rate for each parameter. Adam uses two more values to update the parameters of the model; an estimate of the first moment (mean) and the second moment (uncentered variance) of the gradients. During this research Adam was used with the default parameters of TensorFlow (learning rate = 0.001, β_1 = 0.9, β_2 = 0.99, ϵ = $1e^{-8}$) for all models including the Neural Click Model which is used to compare the proposed models.

Chapter 3

Background

This section describes related work on click models, both based on the PGM framework as well as based on a Deep Learning approach. The dataset used in this project is also discussed along with some statistics.

3.1 PGM based models

The PGM framework allows for very simple models as well as very complex models.

PGM based click models often split the probability of clicking a document into a probability of seeing the document (the examination probability), the probability of a document being attractive and the probability of a document being relevant. The difference between relevance and attractiveness is that relevance describes the content of a document, whereas attractiveness describes the appeal of the snippet in a SERP.

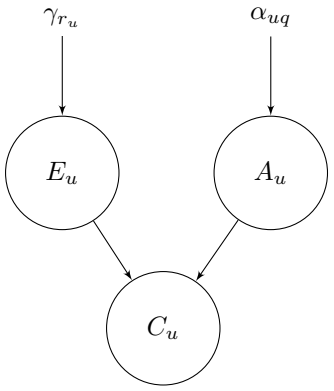


Figure 3.1: Position-based model

Figure 3.1 shows the Position-Based Model (Craswell et al., 2008). The parameter γ_{r_u} is a probability based on the rank r that describes how likely it is that document u on rank r will be examined. The parameter α_{uq} is the probability of document u, shown as a result for query q, being attractive. Both these parameters can be learned from data.

In the same paper Craswell et al. (2008) presented the Cascade Model (Figure 3.2). This model works under the assumption that a user browses the SERP from top to bottom and that the examination probability depends on whether or not the user has clicked the previous document. A document can only be examined if none of

the previous documents have been clicked.

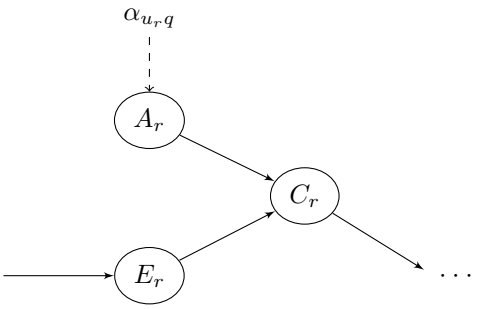


Figure 3.2: Cascade Model

The User Browsing Model (Dupret and Piwowarski, 2008) also takes previous clicks into account when calculating the examination probability. In this model the examination probability is determined by the current rank and the rank of the previous clicked document. Figure 3.3 shows a graphical representation of this model.

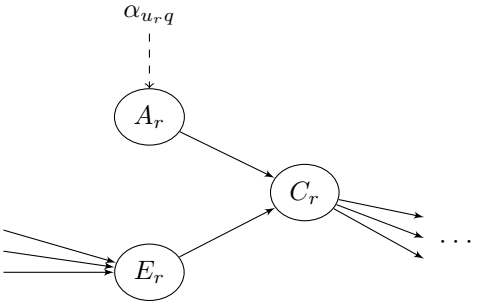


Figure 3.3: User browsing model

The dependent click model (Guo et al., 2009b) is also an extension of the Cascade Model. This model, shown in Figure 3.4, can handle multiple clicks unlike the Cascade Model. In addition to the examination probability and the click probability, this model also has a satisfaction probability (S). The satisfaction describes whether or not the document has sufficiently satisfied the users information need. The satisfaction probability depends on the probability of the document being clicked and the continuation parameter.

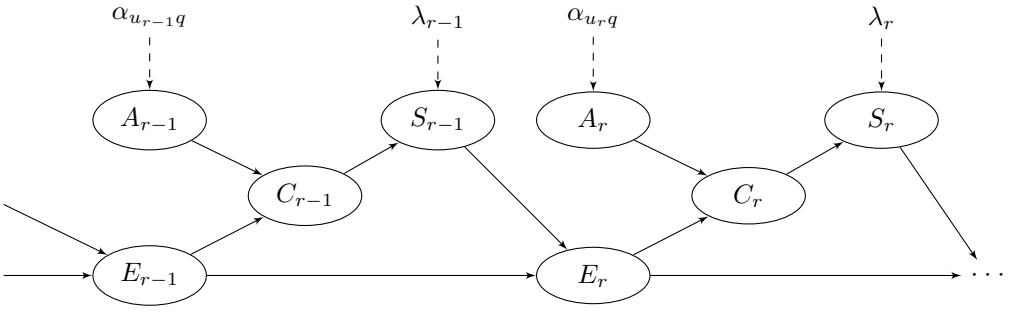


Figure 3.4: Dependent Click Model

The continuation parameter in the Dependent Click Model is based on the rank. In the click chain model (Guo et al., 2009a) the satisfaction depends on the attractiveness of the document. CCM also introduces new parameters (continuation parameters) which allows for predicting a sequence where a user abandons the SERP without clicking any documents. Figure 3.5 shows a graphical representation of the model.

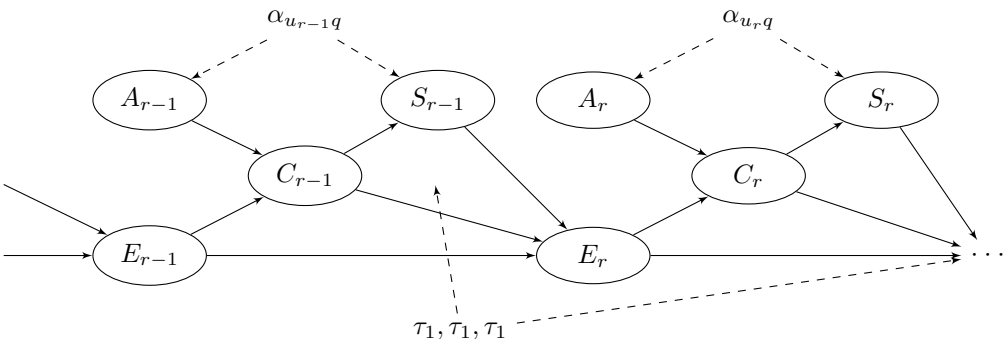


Figure 3.5: Click Chain Model

The Dynamic Bayesian Network Model (Chapelle and Zhang, 2009) introduces the notion of relevance (σ_{uq}). It also extends the Cascade Model, but whether or not a document satisfies a user's information need depends on the attractiveness of the document it depends on the relevance of the document. There is also a continuation parameter that allows the model to predict the user abandoning the SERP without clicking a single document. Figure 3.6 shows the model.

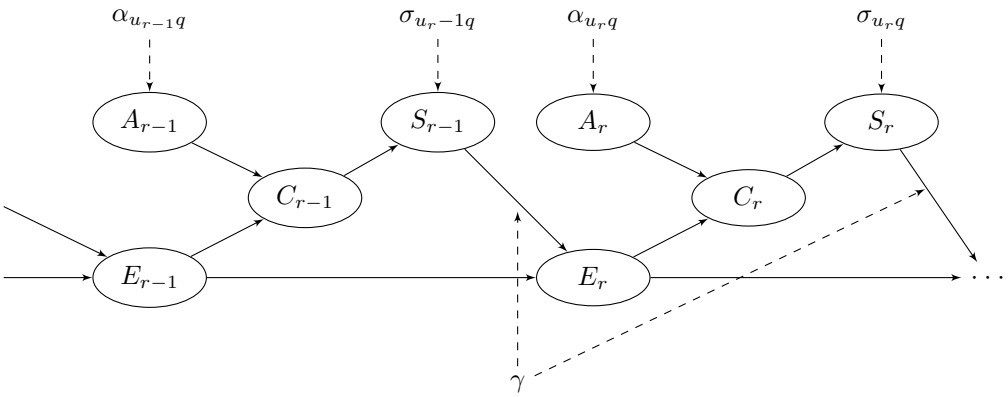


Figure 3.6: Dynamic Bayesian Network Model

All the models in this section and a few more are explained in more detail by Chuklin et al. (2015, chap. 3,8). The important thing to note about these models is that most of them are an extansion of the cascade model. This means that they assume that the user examines the results in the SERP from to bottom.

3.2 Neural Click Model

The neural click model, introduced by Borisov et al. (2016), is a click model that is based on a neural network. In their paper multiple configurations of the network, as well as multiple types of features, are discussed and evaluated. Only the best performing configuration and features are explained here and used for testing.

The features that this model uses are divided into the query representation (q), the document representation (d) and the interaction representation (i).

- **q**: The query representation consists of all possible click patterns on SERPs generated by the query. There are $2^{10} = 1024$ possible click patterns, therefore the representation consists of 1024 numbers, each representing the amount of times that click pattern was observed with this particular query.
- **d**: The document representation consists of two parts. The first part is a representation of all the possible click patterns where the document can be on any of the ranks, regardless of this query. this leads to $10 \times 2^{10} = 10240$ values. The second part consists of all the possible click patters where the document can be on any of the ranks, but only with a specific query. This leads to another 10240 values, bringing the total size of the document representation to 20480 values.
- **i**: The interaction representation is a binary vector of size 1, where the value 0 indicates the the document on the previous rank was not clicked, while a value of 1 indicates that the document on the previous rank was clicked.

This leads to a feature vector of size $1024 + 20480 + 1 = 21505$ for each document in the SERP.

The model that performed best consists of a Long Short-Term Memory (LSTM) network. The LSTM cell is randomly initialized, but the information from the query is added by first passing the query representation concatenated with zeros of the size of the document representation and of the interaction representation through the LSTM. Then the LSTM is passed the features for each document consecutively, giving a hidden state as output for each document. The final step is passing these hidden states through a fully connected layer with a tanh or sigmoid activation function to obtain a prediction for each document. This model is shown in Figure 3.7

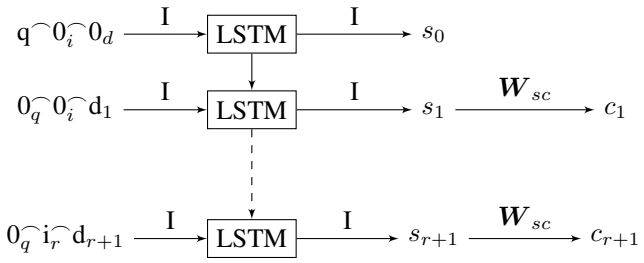


Figure 3.7: Neural click model (Borisov et al., 2016)

3.3 Data

The Yandex Relevance Prediction dataset is used in this project. This is a large dataset containing over 30 million unique queries and over 117 million unique documents. The dataset contains information about search sessions where each session has a unique ID and consists of one or multiple queries. The dataset also keeps track of the time between events in a session, allowing for ordered click sequences to be extracted from the dataset.

Wang et al. (2015) found that, depending on the dataset, about 30% of the multiclick sequences are non sequential.

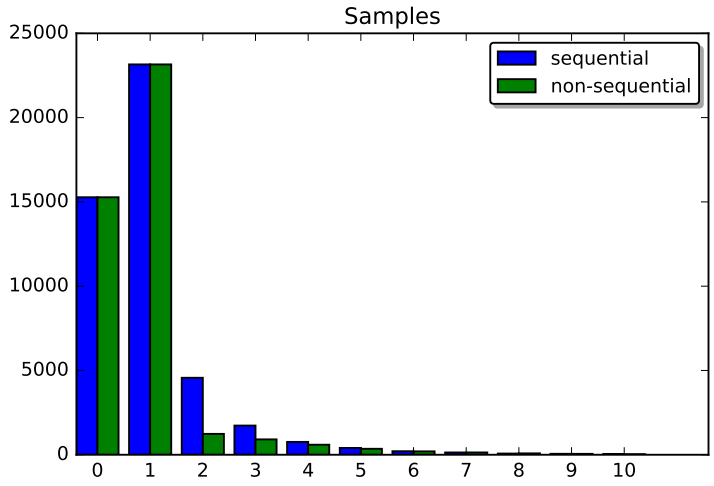


Figure 3.8: Figure showing the distribution of sequences by length and whether they are sequential or non sequential. Note that the sequential and non sequential values for sequences of length zero and one are the same, since these cannot be (non)sequential.

Figure 3.8 is created from a sample of 50.000 query sessions taking from the training set of the Yandex Relevance Prediction dataset. This sample is also the test set that was used to evaluate the models, the results of which can be found in section 6. Table 3.1 shows the same numbers in a more detailed manner. What can be seen is that the data is very skewed towards shorter sequences. It is also clear that there are significantly more sequential sequences than non sequential sequences. Note that the number of sequences of length 0 and 1 are the same for sequential and non sequential, since a sequence of length ≤ 1 cannot said to be sequential or not.

3.4 Evaluation Metrics

The following evaluation metrics are often used in the field of Information Retrieval to evaluate the perfomance of click models.

Sequence Length	Sequential	Non sequential
0	15275	15275
1	23156	23156
2	4566	1238
3	1731	912
4	762	595
5	404	353
6	209	201
7	135	136
8	71	82
9	56	53
10	40	25
Total	7974	3595

Table 3.1: Table showing the distribution of sequences by length and whether they are sequential or non sequential. Note that the sequential and non sequential values for sequences of length zero and one are the same, since these cannot be (non)sequential.

3.4.1 Likelihood

The likelihood is a measurement that describes how well the data fits a probabilistic model. The idea behind it is, instead of making predictions and then comparing it to the test data, to take the test data and to calculate how likely it is to encounter this data given the model.

$$\mathcal{L}(\theta|x) = \prod_{i=1}^{N} P(x_n|\theta) * y_n \tag{3.1}$$

Equation 3.1 shows that the likelihood function is a product over predictions where the click probability is multiplied with the label, thus multiplying the probabilities of those events that actually happened according to the data. The likelihood functions is also used as the cost function. It is common to use the logarithm of the likelihood (Equation 3.2), which offers computational advantages, such as an easier derivative and more efficient calculations since it uses a sum instead of a product.

$$\mathcal{LL}(\theta|x) = \sum_{i=1}^{N} log\big(P(x_n|\theta) * y_n\big) \tag{3.2}$$

3.4.2 Perplexity

Perplexity (Dupret and Piwowarski, 2008) is a metric that describes how 'surprised' a model is by encountering the data. The key is to calculate how likely it is that a document is clicked on a particular rank and, in the case of models that do not assume linear examination, on a particular timestep.

$$p_r(M) = 2^{-\frac{1}{|S|}\sum_{s\in S} log_2 P_M(C_r=c_r^{(s)})}, \tag{3.3}$$

where S are the sessions, and P_M is the probability of a click on event $c_r^{(s)}$ at rank r in session s predicted by the model M. For sequential models there are $2^{10} = 1024$ possible sequences when considering SERPs with ten documents.

For models that do not assume linear, top to bottom examination the element of time also has to be taken into consideration. During this project 10 timesteps are considered for each SERP, while in each timestep the user can either click on any of the ten documents or not click at all. Calculating the probabilities for all these possible sequences is, with the hardware and time constraints, intractable. Therefore only the first three timesteps are considered when calculating the perplexity, with the additional constraint that the user can only click once on each document. This leads to 821 possible sequences, where the perplexity is calculated as shown in Equation 3.4.

$$p_{r,t}(M) = 2^{-\frac{1}{|S|}\sum_{s\in S} log_2 \sum_{t\in T} P_M(C_{r,t}=c_{r,t}^{(s)})} \tag{3.4}$$

Chapter 4

Models

Both models proposed in this research are extensions of the model by Borisov et al. (2016). The proposed models are called the non-sequential neural click model (NSNCM) and the non-sequential click model with attention (NSNCMA). The representations for the queries and documents are the same as described in the Neural Click Model paper and can also be found in section 3.2. Figure 4.1 shows an overview of the non sequential click models. First two LSTMs are initialized with the user query, after which both LSTMs go over the representations of the documents to update the vector state; one LSTM goes from top to bottom and the other from bottom to top. This process is called encoding, and is further explained in section 4.1.

The information of these LSTMs is concatenated and used during the second phase of the algorithm, called the decoder. This phase produces the predictions based on information of the encoder and the user interactions. The encoder is the same for both non sequential models, while the decoder differs between the NSNCM and NSNCMA. The decoders for NSNCM and NSNCMA are explained in sections 4.2 and 4.3 respectively.

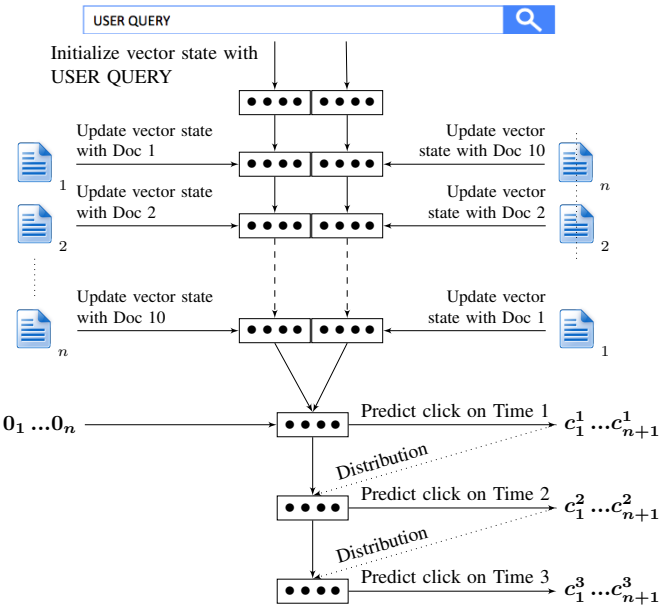


Figure 4.1: Predicting ordered click sequences on a SERP using a non-sequential neural click model.

4.1 Encoder

The aim of the encoder is to obtain information from the query and documents and pass this information to the decoder. The encoded information of the documents should contain information about the document and the documents surrounding it, therefore a bidirectional LSTM is used to create the encodings for the documents; one processing the documents from top to bottom and one processing the documents from bottom to top. The resulting encodings are then concatenated so that the encoding of document d_r contains information of that document and the documents above ($d_{<r}$) and below ($d_{>r}$) it in the ranking. Both LSTMs are first passed the representation of the query concatenated with a zero vector of the size of the document representation. The documents are passed to the LSTMs by concatenating a zero vector of the size of the query representation with the document representations. The LSTM size has been set to 256 during this research.

The output of the encoder consists of three tensors: the concatenated final cell states of both LSTMs ($\boldsymbol{c}$), the concatenated final hidden states of both LSTMs ($\boldsymbol{h}$) and a three dimensional tensor that has a concatenated hidden state from both LSTMs for each document ($\boldsymbol{s_i}$). An overview of the encoder can be found in Figure 4.2.

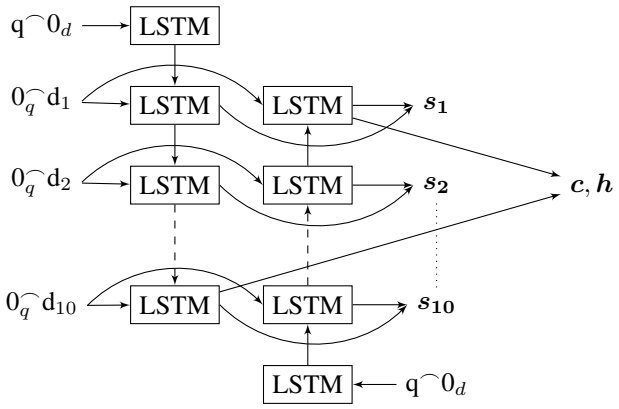


Figure 4.2: Encoding the information about the query and the documents.

4.2 Decoder: Non sequential neural click model

The key behind the non-sequential neural click model (NSNCM) is that it predicts probabilities of events happening on timesteps, instead of only predicting which events will happen. The PGM based models and the neural click model that are discussed in section 3 give a probability to each event (a click on a certain document) for a given SERP. This model outputs a probability distribution on each timestep over all possible events. The possible events are clicking on one of the ten documents or not clicking any document, thus there are 11 possible events. The output therefore consists of a $n \times 11$ matrix where each row is a probability distribution of a certain timestep and each column represents a certain event.

The decoder consists of a LSTM and a fully connected layer with a softmax activation function to produce the probability distributions. A LSTM is used since the previous interactions could influence the current behavior of the user. The LSTM is initialized using the memory state ($\boldsymbol{c}$) and the hidden state ($\boldsymbol{h}$) that were produced

during the encoding (see Figure 4.2). A projection layer ensures that the the output of the encoder LSTM fits the cell and memory state of the final LSTM.

The input to the LSTM are user interactions on the previous time step. The first input is always a zero vector of the size of the interaction vector; the size of the interaction vector is determined by the number of returned documents plus one, since there is always the option of not clicking on any document. A LSTM size of 128 was used during this research.

Finally the probability distribution is calculated from the hidden states produced by the LSTM network. Equation 4.1 shows how the final distribution is calculated. The matrix W_{hc} allows for a linear transformation of the hidden state vectors to the size of the output vectors.

$$c_r = softmax(W_{hc} \cdot h_t) \tag{4.1}$$

The softmax function serves to ensure that the output is a probability distribution. By definition it produces a vector where each element is in the range [0,1] and the sum of the values is 1. The softmax function can be seen in equation 4.2.

$$softmax(x) = \frac{e^x}{\sum_i^N e^{x_i}} \tag{4.2}$$

Figure 4.3 shows a schematic of the entire non-sequential neural click model. The top part represents the encoding of the documents and the bottom part shows how the information from the documents and queries are used to predict an ordered click sequence.

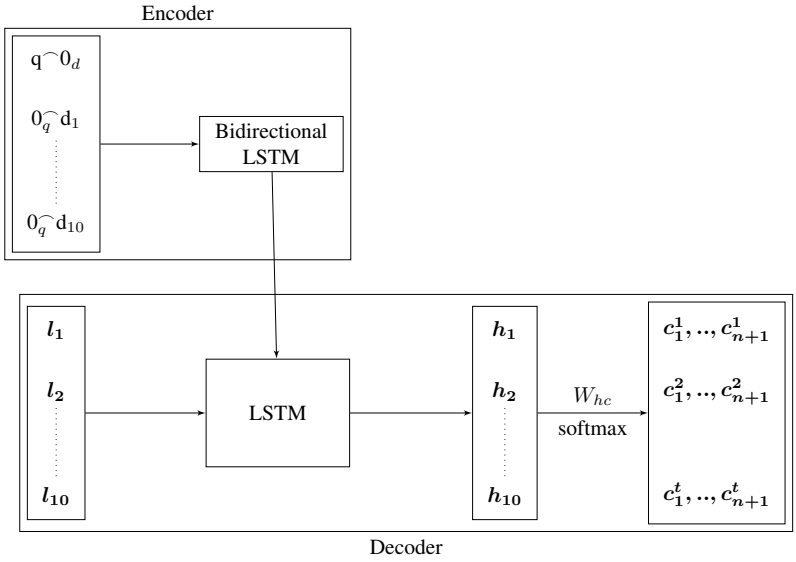


Figure 4.3: Encoding the information about the query and the documents.

4.3 Decoder: Non sequential neural click model with attention

This model is an extension of the non sequential click model discussed in the previous section. The difference between the models is that this model contains an attention mechanism inspired by the work of Bahdanau et al. (2014). It is therefore called the Non Sequential Neural Click Model with Attention (NSNCMA). The intuition behind using the attention mechanism is that by allowing the decoder to look back at the encodings of the documents the long distance dependencies are better captured than when only the LSTM initialisation has information about the documents. The encoder works the same as the regular non sequential click model. However, the encoder does not only output the final cell and memory states but also the memory state for each document (s_n in figure 4.2).

In the decoding step the LSTM uses the labels as input and produces a hidden state for every timestep. The size of this LSTM was set to 256 during this research. The hidden states for the documents are used to create a context vector. This vector is created by multiplying the hidden state from the final LSTM with the hidden state from the documents. A softmax layer over this multiplication creates a context vector that sums to one. The idea behind this method is that the context vector is a weighted sum over the documents, thus providing context to the final LSTM which can be used to better predict the probabilities on clicking the documents.

The context vector is concatenated to the hidden state of the LSTM. The resulting vector is put through a single layer to produce the probability distributions. Figure 4.3 shows a diagram of the model.

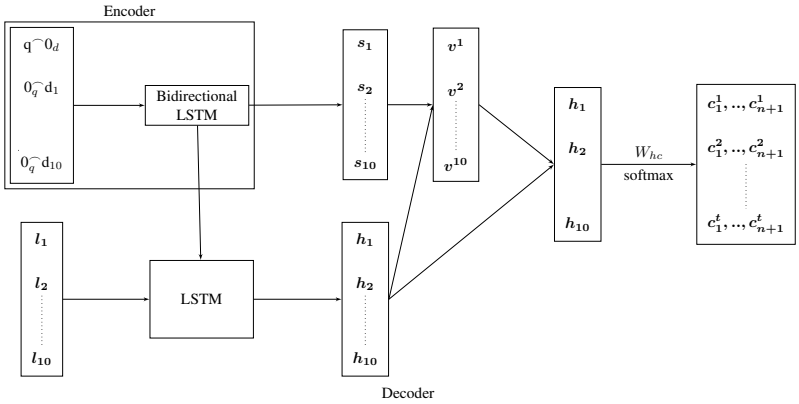


Figure 4.4: Diagram of the Non Sequential Neural Model with Attention (NSNCMA).

Chapter 5

Evaluation Metrics

The evaluation metrics explained in section 3.4 are frequently used metrics for measuring the performance of click models. The problem with these metrics is that they cannot be used to measure the difference in performance of the models in this work, since the output is too different. The likelihood is not comparable since the output of NCM concists of 10 probabilities (a click or not on every document) where a random guess would be 0.5 for each document. For the new models, however, a random guess would be $\frac{1}{11}$ for every document on 10 timesteps. As will be shown in section 6 this leads to an unfair advantage for NCM over the proposed models. The difference in output also makes the perplexity an unfair metric. This section introduces two new evaluation metrics; the Damerau-Levenshtein Distance and the Longest Common Subsequence. These metrics are based on existing algorithms and can be used to evaluate the performance on sequential and non sequential sequences since they are sensitive to order. The Reciprocal Rank is not a new evaluation metric but it is, as far as we know, not frequently used for this task.

5.1 Longest common subsequence

The longest common subsequence (LCS) is measured between two strings or arrays. As the name suggests, the idea behind this metric is to measure what the longest subsequence is that is shared between both strings or arrays.

source	target	LCS	Normalised
[1, 3, 4, 2]	[1, 2]	2	$2/4$
[1, 2]	[1, 3, 4, 2]	2	$2/4$
[1, 3, 2, 4]	[1, 2, 3, 4]	3	$3/4$

Table 5.1: Examples of LCS

Table 5.1 shows three examples of the longest common subsequence. The first two show that it doesn't matter which of the arrays is the correct one, the longest common subsequence is the same. The last example shows that the longest common subsequence is indeed sensitive to order. The longest common subsequence is either [1, 3, 4] or [1, 2, 4], and the length is 3 in both cases. Lastly the LCS is normalised by dividing the LCS by the maximum of the lengths of the source and target, to penalize candidates that contain too many or too few clicks.

The score is calculated by going through the N best predictions and calculating the LCS for each prediction. The final score is then calculated using the Discounted Cumulative Gain (DCG) which gives a single score for the N best predictions. Since the maximum score of the normalised LCS is 1.0 en the minimum is 0, for the

DCG also goes that a higher DCG means better performance.

$$DCG = \sum_{i=0}^{N} \frac{LCS_i}{log_2(i+1)} \tag{5.1}$$

5.2 Damerau-Levenshtein distance

The Damerau-Levenshtein distance is a metric that is based on work of Damerau (1964) and Levenshtein (1966). The Levenshtein distance between two arrays is calculated by counting the number of operations that are needed to go from one array to the other. The operations that can be used are deletion, addition and substitution. Table 5.2 shows an example of the Levenshtein Distance.

Source	Target	Operation
[1, 3, 2, 4]	[1,5]	Initial
[1, 2, 4]	[1,5]	Delete 3
[1, 4]	[1,5]	Delete 2
[1, 5]	[1,5]	Substitute 4 for 5

Table 5.2: Example of Levenshtein Distance

The Damerau-Levenshtein distance adds another operation; transposition. Transposition allows for the swapping of two adjacent elements in the array.

Source	Target	Operation
[1, 3, 2, 4]	[4,1]	Initial
[1, 2, 4]	[4, 1]	Delete 3
[1, 4]	[4,1]	Delete 2
[4, 1]	[4, 1]	Swap 4 and 1

Table 5.3: Example of Damerau-Levenshtein Distance

One of the advantages of the (Damerau-)Levenshtein Distance is that it is possible to assign different penalties for the operations. Though the aim of this project is to predict the ordered sequence of clicks, it is intutively still more important to predict the right clicks than the right order. Therefore the penalty for transposition is set at 0.75, while the penalty for the remaining three operations is set at 1.0. The example in Table 5.3 would have a Damerau-Levenshtein distance of 2.75 while the example in Table 5.2 has a Damerau-Levenshtein distance of 3.0.

The final score is then obtained by normalising the distance by using equation 5.2.

$$DL\_distance = \frac{1}{\#operations + 1} \tag{5.2}$$

The DCG is then calculated in the same way as the LCS. The highest obtainable normalised distance is 1 and the lowest is 0, therefore the DCG scores are to be interpreted as higher is better.

5.3 Reciprocal rank

To measure the quality of retrieving the correct sequence the reciprocal rank is used. The testing algorithm retrieves the N (this research uses N=10) best sequences and orders them by likelihood. The algorithm then loops through the list of predictions and compares them to the actual sequence, if a correct sequence is found the reciprocal rank is calculated as $\frac{1}{rank}$. If the correct sequence is not part of the predicted sequences, the algorithm returns a default score of $\frac{1}{N+1}$.

Chapter 6

Experiments

This section first explains how the experiments were performed and then shows the results that were obtained from the experiments.

6.1 Experimental setup

The three neural click models explained in sections 3.2 (NCM) and 4 (NSNCM and NSNCMA) are implemented using the TensorFlow library. The NCM was implemented with a LSTM size of 256, a sigmoid activation function and the default TensorFlow parameters of the Adam optimizer (section 2.3). NSNCM has a LSTM size of 256 for the encoder, a LSTM size of 128 for the decoder, a Softmax activation function and also the default TensorFlow parameters of the Adam optimizer. NSNCMA has the same configuration as NSNCM, except for the size of the LSTM of the decoder which is of size 256.
The Yandex Relevance Prediction dataset (section 3.3) is split into a training and a test set. After shuffling the data, these sets are divided into smaller files containing one million query sessions each. For training all models use the first 30 files, thus using 30 million query sessions in total. During testing all models use the first 50.000 query sessions from the same test file.
During the experiments both the common metrics (section 3.4) as well as the proposed metrics (5) are used.

All training and testing was done using the same hardware:

- Intel Xeon E3-1231v3
- 16GB DDR3 RAM
- Nvidia GeForce 1060 6GB

6.2 Likelihood

The likelihood is measured as explained in section 3.4.1. Since we are interested in comparing sequential and non sequential models, the results of the likelihood are divided between likelihood for sequential sequences and non sequential sequences. Another reason that doing this is important is due to the fact that there are significantly more sequential sequences in the data than non sequential sequences. Figure 6.1 shows the likelihood of the three neural click models on sequential sequences. The likelihood is an unfair metric since the

probability distributions of the sequential model and the non sequential models are different. Since the non sequential models produce probability distributions over 11 possible events on 10 timesteps while the sequential model produces 10 probabilities of a single event happening, the sequential model has an advantage over the non sequential models. Due to the fact that the sequential model cannot output non sequential predictions, the sequential model calculates the likelihood of non sequential click sequences as if they were sequential. This means that the values in Figure 6.2 cannot be used to compare the sequential model to the non sequential models.

Figure 6.1 shows that the sequential model clearly outperforms the other two models. The difference in scores between the non sequential models is minimal, with NSNCMA having but a slight edge over NSNCM.

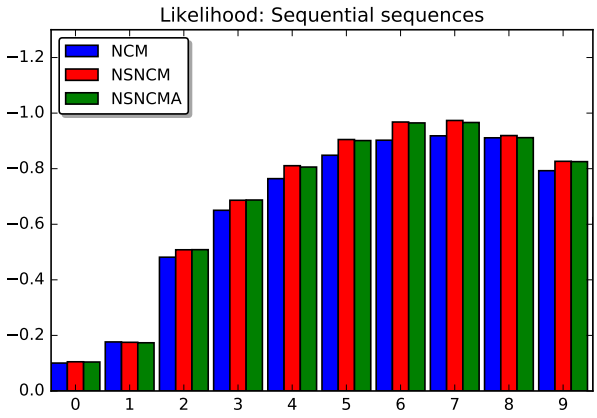


Figure 6.1: The average likelihood of all models calculated over the sequential sequences per sequence length.

Figure 6.2 shows the likelihood of the three models on non sequential sequences. Again there is no significant difference between NSNCM and NSNCMA.

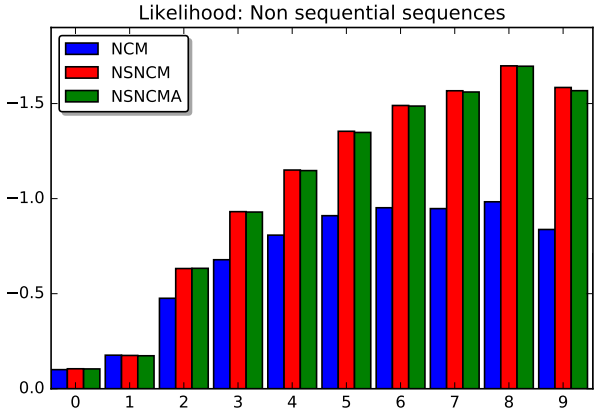


Figure 6.2: The average likelihood of all models calculated over the non sequential sequences per sequence length.

6.3 Perplexity

The perplexity is, like the likelihood, not a fair metric to compare sequential models to non sequential models. Due to restraints in time and computational power the non sequential models only predict up to three timesteps, and always assume a skip in the remaining timesteps. Doing this makes the computation of the perplexity tractable, but also causes the non sequential models to underestimate themselves.

Table 6.1 shows the mean perplexity of the three models. The non sequential model with attention is slightly better than the non sequential model without attention, but both are clearly outperformed by the regular neural click model.

Model	Mean Perplexity
NCM	1.3342
NSNCM	1.3398
NSNCMA	1.3376

Table 6.1: Mean perplexities of the three neural models.

Figure 6.3 shows the perplexity of the three models on sequences of length 0 and 1 and sequential and non sequential sequences. The models are on par on the sequences of lengths 0 and 1. On longer sequences the regular neural click model appears to perform better.

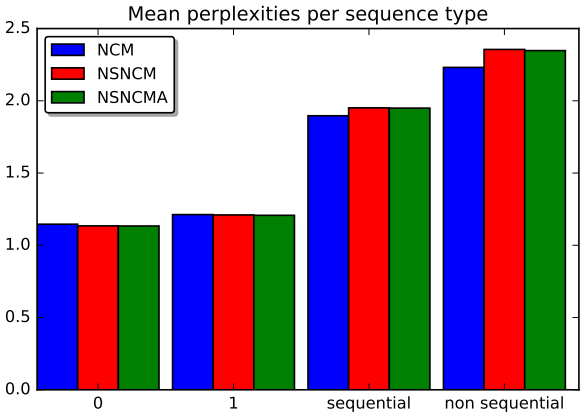


Figure 6.3: The average perplexity of the models on sequences of length 0 and 1 and sequential and non sequential sequences of length ≥ 2

6.4 Reciprocal Rank

To measure the reciprocal rank, the results are split into sequences of length 0, sequences of length 1, sequential sequences of length > 1 and non sequential sequences of length > 1. Figure 6.4 shows the reciprocal rank of the regular and both the non sequential models on sequence of length 0 and 1, sequential sequences and non sequential sequences.

The regular neural click model outperforms the non sequential model on sequences of length 0, but is overtaken by the non sequential models on sequences of length 1. NCM performs slightly better on sequential sequences of length ≥ 1 than the non sequential models, but not in a significant way. On non sequential sequences the non sequential models score better than NCM, which is to be expected since NCM cannot predict non sequential sequences at all. Note that $\frac{1}{11}$ is the lowest achieveable score.

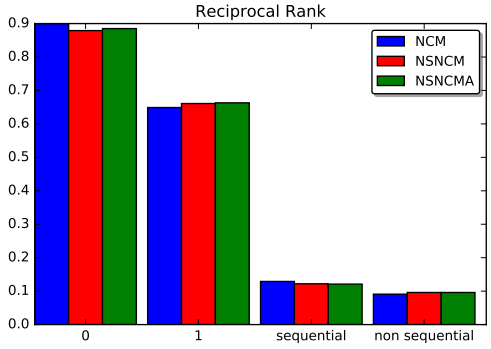
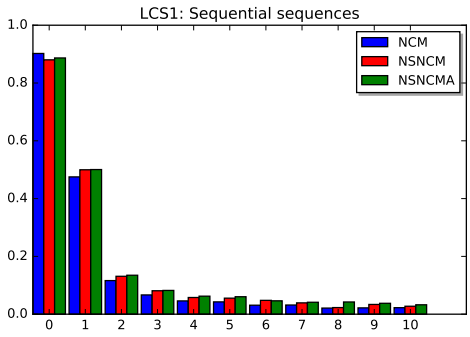


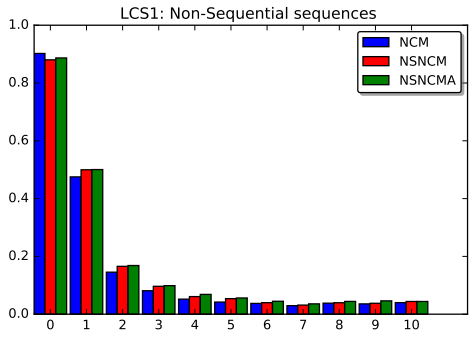
Figure 6.4: Reciprocal rank on sequences of length 0, 1, sequential sequences of length ≥ 1 and nonsequential sequences of length ≥ 1

6.5 Longest common subsequence

The longest common subsequence is presented as DCG scores measured at ranks 1, 3, 5 and 10. Figure 6.5 shows the DCG@1 scores of the LCS on the sequential sequences and non sequential sequences. The results on the sequences of length 0 and 1 are the same for both types of sequences. NCM outperforms NSNCMA which in turn outperforms NSNCM for sequence without clicks. Sequences of length 1 are predicted better by NSNCM(A) which perform about equal. For sequences of lengh ≥ 2 NSNCMA scores better than NCM and NSNCM. Note that NSNCMA also outperforms NCM on sequential sequences.



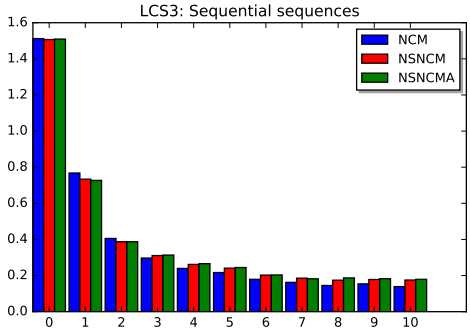
(a) LCS DCG@1 of sequential sequences



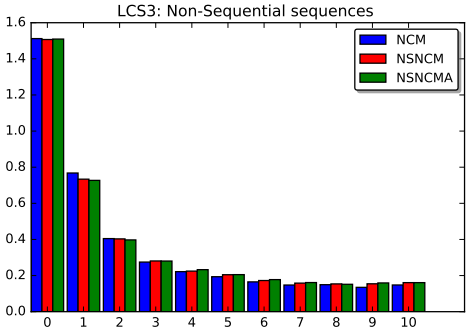
(b) LCS DCG@1 of non sequential sequences

Figure 6.5: LCS DCG@1 on a) sequential sequences and b) non sequential sequences.

Looking at the DCG@3 (Figure 6.6) the results are quite different. The advantage of NCM over NSNCM(A) is smaller, but here NCM performs better on sequence of length 1 and 2. For sequences of length ≥ 3 NSNCMA shows the best performance. NSNCM only performs best by a very little margin on sequential sequences of length 7 and non sequential sequences of length 8.



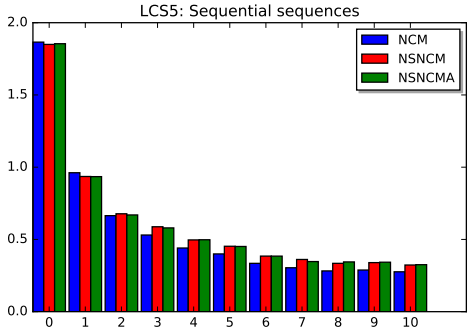
(a) LCS DCG@3 of sequential sequences



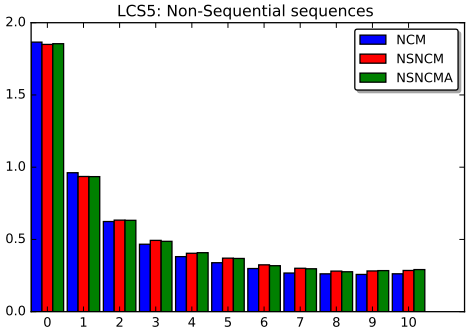
(b) LCS DCG@3 of non sequential sequences

Figure 6.6: LCS DCG@3 on a) sequential sequences and b) non sequential sequences.

Figure 6.7 shows again that NCM has the best performance on sequences of length 0 and 1. However, the difference in performance in favor of NSNCM(A) on longer sequences increases and NSNCM(A) outperforms NCM starting from sequences of length 2.



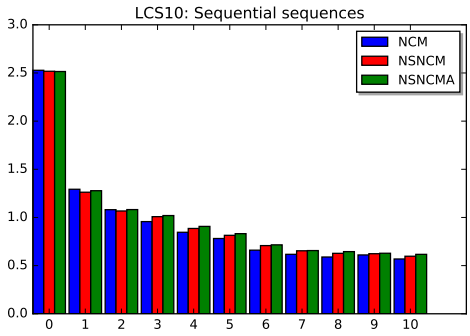
(a) LCS DCG@5 of sequential sequences



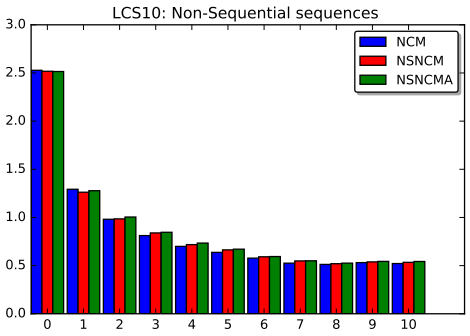
(b) LCS DCG@5 of non sequential sequences

Figure 6.7: LCS DCG@5 on a) sequential sequences and b) non sequential sequences.

NSNCMA takes the lead in performance of sequences of length 3. Other than that the same trend is found at rank 10 as at rank 5.



(a) LCS DCG@10 of sequential sequences

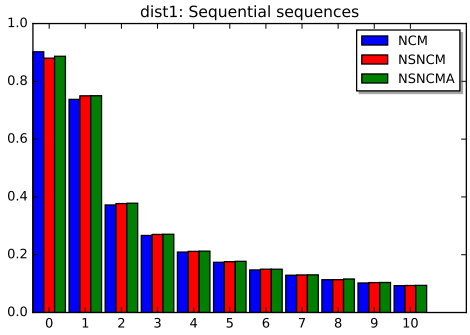


(b) LCS DCG@10 of non sequential sequences

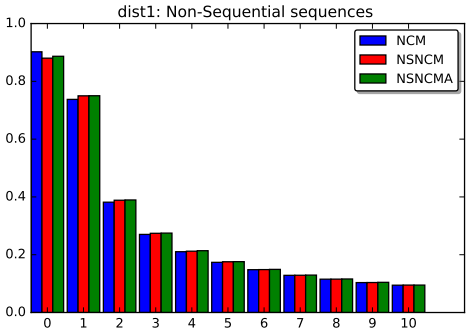
6.6 Distance

The Damerau-Levenshtein distance is presented as DCG scores measured at ranks 1, 3, 5 and 10. The figures are divided into sequential and non sequential sequences. Note that the sequence lengths of 0 and 1 are the same for both type of sequences since these are not sequential or non sequential by definition.

Figure 6.9 shows that NSNCMA outperforms NCM and NSNCM on all sequence lengths of size ≥ 1. NCM only manages to outperform the non sequential models on sequences of length 0. Though NSNCMA outperfms NSNCM, the differences are negligeable except for sequences of length 0.



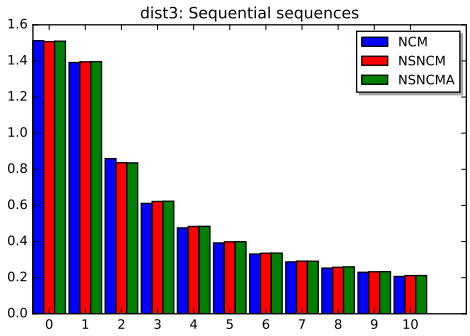
(a) Distance DCG@1 of sequential sequences



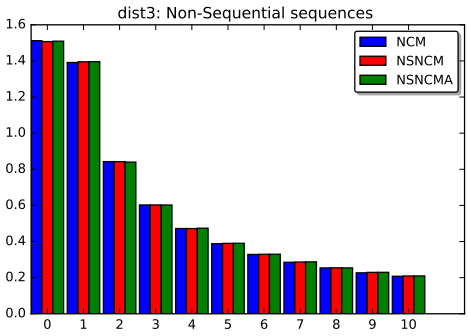
(b) Distance DCG@1 of non sequential sequences

Figure 6.9: Damerau-Levenshtein distance DCG@1 on a) sequential sequences and b) non sequential sequences.

At DCG ranks 3 and 5 the difference in performance between the models on sequence of lengths 0 and 1 are decreased, as can be seen in Figure 6.10 and Figure 6.11. NCM outperforms the non sequential models for sequential sequences of length 2, but still scores less than the non sequential models on longer sequential and non sequential sequences.

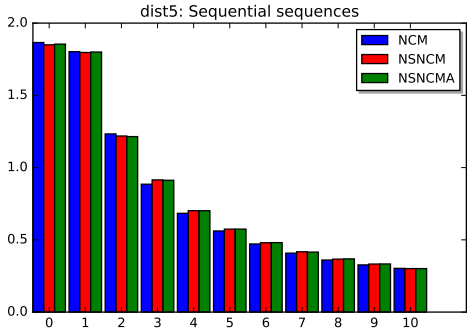


(a) Distance DCG@3 of sequential sequences

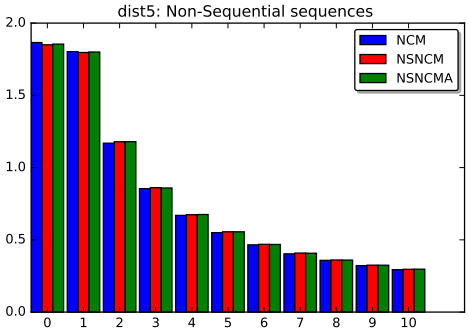


(b) Distance DCG@3 of non sequential sequences

Figure 6.10: Damerau-Levenshtein distance DCG@3 on a) sequential sequences and b) non sequential sequences.



(a) Distance DCG@5 of sequential sequences



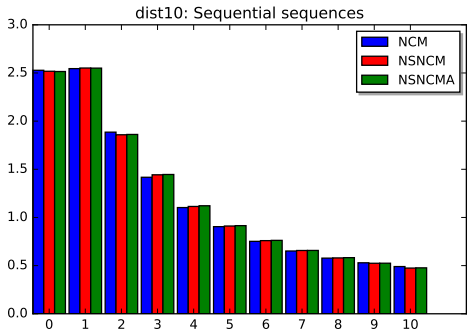
(b) Distance DCG@5 of non sequential sequences

Figure 6.11: Damerau-Levenshtein distance DCG@5 on a) sequential sequences and b) non sequential sequences.

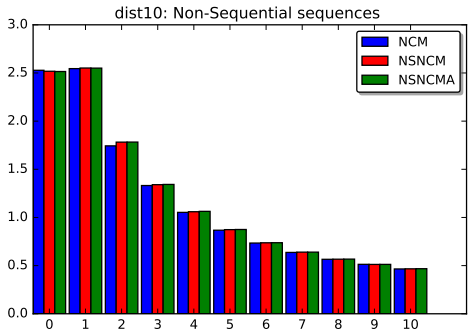
Figure 6.12 shows the scores on the DCG@10. Interesting here is to see that the the performance of all three models on both sequential and non sequential sequences of length 1 are better than on sequences of length 0. Other than that we see the same trend as on ranks 3 and 5, where NCM performs best on sequential sequences of length 2 and the non sequential models perform better on non sequential sequences of length 2 and all sequences of length ≥ 3. The difference between the non sequential models is very slim, with NSNCMA having a slight edge over NSNCM.

6.7 Prediction lengths

The majority of the search sessions in the data has a click sequence of length 0 or 1. Since neural networks are susceptible to skewed data, it is interesting to find out what the bias of the models is towards shorter sequences. Table 6.2 shows that in the test data that was used to evaluate the models over 75% of the search sessions had



(a) Distance DCG@10 of sequential sequences



(b) Distance DCG@10 of non sequential sequences

Figure 6.12: Damerau-Levenshtein distance DCG@10 on a) sequential sequences and b) non sequential sequences.

length 0 (30.5%) or length 1 (46.3%). It is therefore interesting to see which of the models has the least bias towards very short click sequences. Table 6.2 shows the ratio of predicted click sequence lengths for each model and the actual sequences (label). It seems that the regular neural click model has a stronger bias to sequences of length 0 than the non sequential neural models. It should be noted that, as is the case with other evaluation metrics, the non sequential models as implemented during this research does not consider sequences of length >3.

Sequence size	Actual % of sequences	NCM	NSNCM	NSNCMA
0	0.305	0.6019	0.504	0.554
1	0.463	0.3979	0.493	0.445
2	0.116	1.2e-3	0	0
3	0.053	0	2e-3	1e-3
4	0.027	0	0	0
5	0.015	0	0	0
6	0.008	0	0	0
7	0.005	0	0	0
8	0.003	0	0	0
9	0.002	0	0	0
10	0.001	0	0	0

Table 6.2: Ratio of lengths of the actual sequences and predictions. Only the predictions with the highers likelihood for every model on every query session was used.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

During this thesis an attempt was made to create a neural click model that is not only able to predict the documents that will be clicked by a user, but also predict in which order they will be clicked. It was not the primary aim to create a model that outperforms existing click models, but to create a model that is at least on par with existing click models and has the potential for predicting ordered click sequences. It has been shown that the task of predicting ordered click sequences is indeed possible by using deep learning. However, the models presented in this research are far from complete. The results show that the non sequential models outperform the sequential neural click model in most metrics that are sensitive to order (Reciprocal Rank, Damerau-Levenshtein Distance and Longest Common Subsequence), though not in a significant way. In terms of likelihood and perplexity the sequential neural click model still outperforms the non sequential neural click models. A promising result is that the non sequential models appear to have less bias towards empty click sequences than the sequential model.

An issue with the non sequential neural click models that has arisen during this research is the fact that making predictions is far more time costly than with the sequential click model. This is caused by the fact that the non sequential neural click models need to take far more possible click sequences into account.

The two models (NSNCM and NSNCMA) that are presented in this paper both consist of an encoder and a decoder, where the encoder obtains information from the query and the documents whereas the decoder is responsible for the actual predictions. The difference between the models is that NSNCMA has an attention mechanism. It is clear from the results in section 6 that the attention mechanism has a positive effect on the performance of the non sequential neural click model.

7.2 Discussion and Future Work

The models presented in this research show promising results. However, more should be done to reach the full potential of non sequential click modelling using Deep Learning. Due to the limitations of hardware and time, the non sequential models are underestimated by only predicting sequences up to a length of three clicks. A suggestion to resolve these limitations is by using beam search, an algorithm used in applications such as speech recognition (Lowerre, 1976), to get better predictions that are no longer restriced by a maximum number of clicks for a sequence.

Due to the considerable training and testing times very little experimenting has been done with hyperparameters such as the LSTM size, the activation functions and the settings for the optimizer. By finding the optimal values for these parameters the performance may still be enhanced. Also the configuration of the neural nets themselves should be experimented with more. Instead of LSTM cells GRU cells (Cho et al., 2014) can be explored, as well as different alignment models for the attention mechanism.

Another point of interest for further studies would be the features that are used. This research was based on the features presented by Borisov et al. (2016) and hold information that is derived from observed click patterns. These observed click patterns did not hold information about the order, creating features from click patterns that do take order into account would lead to significant larger features. There are other user behaviors that could possibly provide additional or better information such as the time spent on the SERP and/or on clicked documents.

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Borisov, A., Markov, I., de Rijke, M., and Serdyukov, P. (2016). A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*, pages 531–541. International World Wide Web Conferences Steering Committee.

Chapelle, O. and Zhang, Y. (2009). A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1–10, New York, NY, USA. ACM.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Chuklin, A., Markov, I., and Rijke, M. d. (2015). *Click Models for Web Search*, volume 7. Morgan & Claypool Publishers.

Chuklin, A., Serdyukov, P., and de Rijke, M. (2013). Click model-based information retrieval metrics. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 493–502, New York, NY, USA. ACM.

Craswell, N., Zoeter, O., Taylor, M., and Ramsey, B. (2008). An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 87–94. ACM.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

Dupret, G. E. and Piwowarski, B. (2008). A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 331–338, New York, NY, USA. ACM.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

Guo, F., Liu, C., Kannan, A., Minka, T., Taylor, M., Wang, Y.-M., and Faloutsos, C. (2009a). Click chain model in web search. Association for Computing Machinery, Inc.

Guo, F., Liu, C., and Wang, Y.-M. (2009b). Efficient multiple-click models in web search. In *WSDM'09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, Inc.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Lorigo, L., Pan, B., Hembrooke, H., Joachims, T., Granka, L., and Gay", G. ("2006"). "the influence of task and gender on search and evaluation behavior using google". *"Information Processing & Management"*, "42"("4"):"1123 – 1131".

Lowerre, B. T. (1976). *The Harpy Speech Recognition System.* PhD thesis, Pittsburgh, PA, USA. AAI7619331.

Luong, T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *CoRR*, abs/1410.8206.

Malkevich, S., Markov, I., Michailova, E., and de Rijke, M. (2017). Evaluating and analyzing click simulation in web search. pages 281–284.

Markov, I., Kharitonov, E., Nikulin, V., Serdyukov, P., de Rijke, M., and Crestani, F. (2014). Vertical-aware click model-based effectiveness metrics. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1867–1870, New York, NY, USA. ACM.

Wang, C., Liu, Y., Wang, M., Zhou, K., Nie, J.-y., and Ma, S. (2015). Incorporating non-sequential behavior into click models. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 283–292, New York, NY, USA. ACM.

Xing, Q., Liu, Y., Nie, J.-Y., Zhang, M., Ma, S., and Zhang, K. (2013). Incorporating user preferences into click models. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1301–1310, New York, NY, USA. ACM.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.