University of Amsterdam

Noise-Aware Click Modelling

Author:
Tom van den Bogaart

Supervisor:
Artem Grotov

A thesis submitted to the University of Amsterdam in partial fulfilment of the requirements for the degree of Bachelor of Science in Artificial Intelligence

June 26, 2015

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Abstract

Click models aim to infer a user-perceived relevance for each search result from click data. Search engines may use this inferred relevance information to improve their ranking function. Most click models assume that every click is a good indication of relevance and do not take into account that there exists noise in clicks: a user might click on urls that turn out to be irrelevant. In this thesis, a click model is implemented that can capture the noise degree in clicks, which in turn determines the quality of a click as an indication of relevance. As such, high quality clicks with less noise can play a more prominent role for relevance inference. To verify if accounting for noise contributes to the prediction of click behaviour, experiments have been conducted that compare the performance of the noise-aware click model with a classical click model. Experimental results demonstrate that the noise-aware click model significantly deteriorates the click prediction in terms of both perplexity and log-likelihood. In order to accurately measure the effectiveness of the noise-aware model, future research should collect a dataset that includes more queries per search session and a higher percentage of noisy clicks. Additionally, more contextual features could be extracted.

Contents

1 Introduction **1**

2 Theoretical Framework **4**

2.1 Background 4

2.1.1 Preliminary 4

2.1.2 Examination Hypothesis 5

2.1.3 UBM Click Model 5

2.2 Noise-Aware Click Model 6

2.2.1 Model Specifications NCM 6

2.2.2 Model Specifications Noise Aware UBM 7

2.3 Approximate Bayesian Inference 8

2.3.1 Bayesian Inference for Click Models 8

2.3.2 ABI Algorithm 10

3 Methodology **13**

3.1 Approach 13

3.2 Inference Method 14

3.2.1 Approximate Bayesian Inference for N-UBM 15

3.2.2 Sum-Product Message Passing for Noise-Aware UBM 20

4 Evaluation **22**

4.1 Experimental Setup 22

4.2 Evaluation Criteria 23

4.2.1 Log-Likelihood 23

4.2.2 Click Perplexity 23

4.3 Experimental Results 24

5 Conclusion **28**

Chapter 1

Introduction

Search engines retrieve relevant information for a submitted query and present the results in a ranked list. Users examine this list on a 'search engine result page' (SERP) and click on urls[1] that are perceived as relevant, thereby generating a click-through log. Every day, industrial search engines record a large amount of such click-through logs, which provide a valuable source of information about the relevance of the clicked result in relation to the corresponding query. Analysing and interpreting such click-through logs are crucial for understanding search behaviour and for improving search performance [1]. This thesis focuses on the modelling of user click behaviour within Web search, which is defined as a click model problem.

By employing click log data, click models aim to predict user click behaviour or to infer the relevance for each search result as perceived by the user, which can be used to improve a ranking function for search engines in order to provide better search results [2]. An arguably intuitive approach to interpret click logs is to utilising statistics that are encoded in the data. A widely used statistic in click models is the click-through rate (CTR) [3]. Given a specific query, the CTR for a query-url pair is measured by dividing the number of clicks on a url by the number of times a url is shown. The presumption is that a higher CTR value means that more users prefer the url in relation to the entered query and thus, that this higher CTR indicates a better user-perceived relevance.

Unfortunately, the CTR cannot be used as an exact measure of url relevance due to *position bias* — search engine results that appear at the top of the SERP may attract

[1]The term 'url' (and its equivalent 'document') is used as acronym for the search result block that consists of title, snippet and url.

more user clicks than results located more to the bottom of the page, despite the fact that the top results may not be the most relevant to the query. This bias has also been observed by eye-tracking experiments in which Granka et al. [4] were the first ones to demonstrate that a user is less likely to examine results near the bottom of a list. Classical click models, such as the User Browsing Model (UBM), have been developed to alleviate the position bias by estimating the probability of examination for each url [3].

Despite their success in addressing the position bias, classical click models have a considerable limitation, since most of these models presume that every click is a reliable indication of relevance of the url in relation the query [3]. Consequently, each click that is recorded in the click logs contributes to the CTR, resulting in an increase of relevance dedicated to a clicked url. Yet, these models do not take into account that a high percentage of clicked results may be be irrelevant to the user [3]. Hence, real click behaviour is often noisy and is therefore not always a reliable indication of relevance inference. As a result, the general approach of classical models is based on a simplified assumption that considers each click as a binary event (i.e. click or not) while neglecting the noise that may be involved.

Recently, Chen et al. [3] proposed a click model called the Noise-aware Click Model (NCM) that complements click data with human judged data in order to develop a noise predictor that characterises the noise degree of a click. This noise degree can be used to indicate the quality of clicks so that high quality clicks (i.e. clicks with less noise) play a more prominent role in relevance inference.

The present study has implemented NCM as an extension of the User Browser Model (UBM) in order to research whether this model can predict the user's click behaviour more accurate than the classical click model UBM. An attempt will be made to provide an answer to the following research question:

RQ1 *How much does the performance of NCM compare with the performance of the classical click model UBM?*

NCM characterises high quality click data by explicitly accounting for noise in user clicks. In addition, Chen et al. [3] demonstrated that NCM outperforms classical click models such as UBM both in ranking performance and in how well it can predict clicks.

Hence, it is expected that NCM will significantly improve the prediction of click behaviour in comparison with UBM.

In order to research the performance of NCM, this study will conduct several experiments. Firstly, the correlation between the amount of human judged data and the performance of click prediction will be analysed. Secondly, the study will analyse which features are most useful for characterising noise in user clicks in order to improve the prediction of click behaviour. As such, the following sub questions will be answered:

RQ1.1 *How does the amount of human judgements ratings correlate to the prediction of click behaviour?*

RQ1.2 *Which features are most useful for predicting click behaviour?*

The thesis will proceed in six sections. The following chapter will provide background information, whereby it will elaborate on previous research on click models. Subsequently, chapter 3 will present the methodology of the study and will give a detailed illustration of the inference process. Finally, chapter 4 will present the results of the conducted experiments and chapter 5 will conclude this thesis with a discussion and conclusion that will attempt to answer the above-mentioned research question as well as the sub questions.

Chapter 2

Theoretical Framework

2.1 Background

The following section will provide an introduction to some of the relevant terminology and previous research on click models, which will be referred to in the subsequent sections of this thesis.

2.1.1 Preliminary

A user starts a *query session* by entering a *query* into the search engine. The search engine will aim to satisfy the user's informational needs by showing M ranked urls, also called *impressions*, in the *Search Engine Result Page* (SERP). A url (or document) in a SERP ranked at position j is indicated by $u_{\phi(j)}$. It is assumed that all urls are indexed, mapped by function ϕ, to represent the url at position j. In this scenario, a user examines the SERP and may decide to click on one, several or none of the presented documents.

Different literature refer to multiple and interchangeable definitions of a user session. In this thesis, a distinction is made between a query session and a *search session*. A query session captures all the user's actions that are related to the submitted query. A search session entails all actions a user performs while completing the search task, which may include one or more query sessions, multiple reformulations of queries and clicks under different queries [3].

Click models treat examinations and clicks on urls in the SERP as probabilistic events

[3]. When a url at position j is examined in a particular query session, this is denoted by a binary random variable $E_j = 1$ and otherwise $E_j = 0$. In the same way, $C_j = 1$ is used to indicate that the url at position i is clicked and $C_j = 0$ if the url is skipped. For this reason, the examination probability and the corresponding click probability for a url at position j is respectively indicated by $P(E_j = 1)$ and $P(C_j = 1)$.

2.1.2 Examination Hypothesis

The examination hypothesis was developed in order to address the position bias. The key assumption of the examination hypothesis is that a displayed url is clicked *if and only if* this url is both *examined* and perceived as *relevant* [2, 3, 5]. In addition, the examination event depends *only* on the position [2]. More specifically, for a particular query q_i and a url $u_{\phi(j)}$ at the j-th position, the examination hypothesis characterises the corresponding click probability for click event C_j as follows:

$$P(C_j = 1 \mid E_j = 0) = 0 \tag{2.1}$$

$$P(C_j = 1 \mid E_j = 1) = \alpha_{i\phi(j)} \tag{2.2}$$

where $\alpha_{i\phi(j)}$ measures the degree of relevance between query q_i and url $u_{\phi(j)}$, which forms the conditional click probability after examination. Under this examination hypothesis, the click probability can be represented as:

$$P(C_j = 1) = \underbrace{P(E_j = 1)}_{position\ bias} \underbrace{P(C_j = 1 \mid E_j = 1)}_{url\ relevance} \tag{2.3}$$

where the click probability can be decomposed into position bias and url relevance. Thus, given the condition that a url is examined $P(E_j = 1)$, the relevance of a url can be interpreted as the CTR. However, whether a user examined a url at a particular position is not observable under the examination hypothesis. Other click models therefore aim to address this issue by making various assumptions to estimate the probability of examination.

2.1.3 UBM Click Model

The User Browsing Model (UBM) proposes an extension of the examination hypothesis. This model assumes that in the current query session, the examination event E_j depends

not only on the position j but also on the latest clicked position [6]. More specifically, it assumes that the probability of examination depends on the distance d from the last click along with the position r in the ranking. UBM introduces the parameter γ_{rd} that computes the probability that url $u_{\phi(r)}$ at position r was examined when the last click of the user was observed at position $r - d$:

$$P(E_r = 1 \mid C_{1:r-1} = 0) = \gamma_{rr} \tag{2.4}$$

$$P(E_r = 1 \mid C_{r-d} = 1, C_{r-d+1:r-1} = 0) = \gamma_{rd} \tag{2.5}$$

$$P(C_j = 1 \mid E_j = 0) = 0 \tag{2.6}$$

$$P(C_j = 1 \mid E_j = 1) = \alpha_{i\phi(j)} \tag{2.7}$$

where $\alpha_{i\phi(j)}$ measures the degree of relevance.

2.2 Noise-Aware Click Model

This section discusses the model specifications of NCM in general and of NCM as its extension of the classical click model UBM.

2.2.1 Model Specifications NCM

The Noise-Aware Click Model (NCM) proposes a method to characterise the noise degree of a user's click by complementing click data with human judged data. As such, a certain amount of click data can be labelled with human ratings in order to define the noise degree. NCM takes the user preference and contextual information into account in order to generalise the degree of noise for all the unlabelled clicks with a binary random variable $N = 1$ for an extremely noisy context or $N = 0$ for a noise-free context. Thus, the noise probability of a url is defined as $P(N_j = 1)$, which is calculated according to the feature values. NCM also introduces the binary random variable $L_j = 1$ to indicate if the j-th url is human-rated as relevant with respect to the query and $L_j = 0$ if not.

According to these definitions, NCM can be formalised as follows:

$$P(N_j = 1) = g(f_1, f_2, ..., f_n) \tag{2.8}$$

$$P(C_j = 1 \mid E_j = 0) = 0 \tag{2.9}$$

$$P(C_j = 1 \mid E_j = 1, L_j = 1, N_j = 0) = \alpha_{i\phi(j)} \tag{2.10}$$

$$P(C_j = 1 \mid E_j = 1, L_j = 0, N_j = 0) = 0 \tag{2.11}$$

$$P(C_j = 1 \mid E_j = 1, N_j = 1) = b \tag{2.12}$$

In NCM, $\alpha_{i\phi(j)}$ indicates the click probability of url j, given the conditions that the user examines this url in a noise-free context and that this url is human rated as relevant with $L_j = 1$. Each user behaviour feature is indicated by f_j. The probability of $N_j = 1$ is estimated by a function of choice, formalised by $g : R^n \rightarrow R$, which maps all features to a value that indicates the noisiness of a context. In the work by Chen et al. [3], the function g is represented as follows:

$$g(f_j) = \Phi(\sum w_j f_j) \tag{2.13}$$

where each user behaviour feature f_j is accompanied by a corresponding feature weight w_j. $\Phi(x) = \int_{-\infty}^{x} N(t; 0, 1)dt$ represents the cumulative distribution function (CDF) of the standard normal distribution, also referred to as the *probit link*[5]. This is used by the model for convenience in the inference process and to ensure that the probability value falls within the interval of [0, 1]. According to NCM, the estimated relevance of a query-url pair $\alpha_{i\phi(j)}$ is defined as the probability of a click in a noise-free context given that a url has been examined by the user.

NCM uses a Bayesian inference method to learn the parameters, which will be further discussed in section 2.3.

2.2.2 Model Specifications Noise Aware UBM

NCM can be considered as a general model that can embrace the assumptions of many click models, as it does not make any assumptions or constraints in estimating the examination probability. Therefore, NCM can be extended to perform under the assumptions of the UBM model. The Noise-Aware UBM (N-UBM) model can be formalised with the

following equations:

$$P(E_r = 1 \mid C_{1:r-1} = 0) = \gamma_{rr} \tag{2.14}$$

$$P(E_r = 1 \mid C_{r-d} = 1, C_{r-d+1:r-1} = 0) = \gamma_{rd} \tag{2.15}$$

$$P(N_j = 1) = \Phi(\sum w_j f_j) \tag{2.16}$$

$$P(C_j = 1 \mid E_j = 0) = 0 \tag{2.17}$$

$$P(C_j = 1 \mid E_j = 1, L_j = 1, N_j = 0) = \alpha_{i\phi(j)} \tag{2.18}$$

$$P(C_j = 1 \mid E_j = 1, L_j = 0, N_j = 0) = 0 \tag{2.19}$$

$$P(C_j = 1 \mid E_j = 1, N_j = 1) = b \tag{2.20}$$

The N-UBM model computes the noise probability $P(N_j = 1)$ as described in NCM, based on the contextual features values f_j with corresponding learnt weights w_j.

If the user happens to be in a noise-free context $N_j = 0$ and the url has a human judged relevant rating of $L_j = 1$, the click probability $P(C_j = 1)$ is determined by the examination probability $P(E_j = 1)$ and the parameter $\alpha_{i\phi(j)}$ that measures the perceived relevance. However, if the url is human-judged with an irrelevant rating ($L_j = 0$) in a noise-free context ($N_j = 0$), the click probability is assumed to be equal to 0.

Additionally, in the case of an extremely noisy context with $N_j = 1$, N-UMB introduces a query specific parameter b that measures the click probability. According to the UBM model, after the user performs an action, the following urls will be examined with probabilities that are related to parameter γ.

2.3 Approximate Bayesian Inference

The following section will first explain how Bayesian inference can be used in click models. Thereafter, this section will provide a elaboration on the ABI method.

2.3.1 Bayesian Inference for Click Models

Bayesian inference is a statistical approach in which Bayes' rule is used to update the parameters (of the underlying probability distribution) for a hypothesis as new data is observed. The Bayes' rule is:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2.21}$$

Since the scaling operation is trivial, the Bayes' rule is also written as:

$$P(A \mid B) \propto P(B \mid A)\, P(A) \tag{2.22}$$

The probabilities and probability distributions in this expression have the following names:

- $P(A)$ is the *prior distribution*. This distribution represents what is known about A before B is observed.
- $P(B \mid A)$ is the *likelihood*. This distribution represents what is known about B conditional on A.
- $P(A \mid B)$ is the *posterior distribution*. This distribution represents what is known about A after observing B.

Generally, a click model $\mathcal{M}$ is parametrised by a set of unknown variables $\theta_1, ..., \theta_n$. The values of these parameters determine the performance of this click model. Bayesian inference can be used to estimate the parameter values based on a series of query sessions.

As stated in the preliminaries, a query session contains M impressions in which $C_j \in \{0, 1\}$ indicates whether the url at position j is clicked. As such, the likelihood of this query conditional on its parameters is defined as:

$$P(C_{1:M} \mid \theta_1, ..., \theta_n) = f(\theta_1, ..., \theta_n) \tag{2.23}$$

where $C_{1:M}$ is a set of M click impressions for that query session and f is a function of the parameters $\theta_1, ..., \theta_n$.

When Bayesian inference is applied, given a query session, the Bayes' rule can be defined as:

$$P(\Theta \mid C_{1:M}) \propto P(C_{1:M} \mid \Theta)\, P(\Theta) \tag{2.24}$$

The first step of Bayesian inference is to assign a prior distribution $P(\theta_k)$ for each unknown parameter $\theta_k \in \Theta$ of the click model, before any query sessions are observed. Subsequently, given the prior distribution, a query session is observed. Then, the likelihood of the observed query session is calculated as a function of the parameter values. Finally, Bayesian inference derives the posterior distribution $P(\Theta \mid C_{1:M})$ — the distribution of

the parameters after observing the data — by multiplying the likelihood by the prior distribution. This is done according to Bayes' theorem, since $posterior \propto prior * likelihood$.

Click models may use Bayesian inference algorithms which apply the Bayes' rule iteratively by loading and processing query sessions one by one in order to learn parameters. This is done by treating the resulting posterior probability as prior probability in the next iteration in order to compute a new posterior probability from a new query session.

2.3.2 ABI Algorithm

Most classical click models such as UBM, use the traditional Expectation Maximisation (EM) method to infer their parameter values, which is computationally expensive [7]. To address this limitation, Zhang et al. [5] proposed the Probit Bayesian Inference algorithm, which will be referred to as the *Approximate Bayesian Inference* (ABI) algorithm in this thesis. The ABI algorithm is a inference method based on Bayesian inference.

Framework

Most click models, parametrised by a set of unknown variables $\theta_1, ..., \theta_n$, make the assumption that $\theta_k \in [0, 1]$ for $k = 1, ..., n$ [5]. The ABI algorithm proposes to connect each parameter θ_k with a new variable $\breve{\theta}_k \in (-\infty, +\infty)$ that is Gaussian distributed via a so-called *probit link*:

$$\theta_k = \Phi(\breve{\theta}_k),\ k = 1, ..., n \tag{2.25}$$

where $\Phi(\breve{\theta}) = \int_{-\infty}^{\breve{\theta}} N(t; 0, 1)\mathrm{d}t$ is the cumulative distribution function (CDF) of the standard normal distribution. As described above, the likelihood function can be formulated as a function f of $\theta_1, ..., \theta_n$. In the case of UBM and N-UBM, the function f is polynomial. That being said, the likelihood function from equation 2.23 can be rewritten as:

$$P(C_{1:M} \mid \breve{\theta}_1, ..., \breve{\theta}_n) = f(\Phi(\breve{\theta}_1), ..., \Phi(\breve{\theta}_n)) \tag{2.26}$$

The *order* of $\Phi(\breve{\theta}_k)$ is determined by its highest order power in equation (2.26). It is assumed that $\breve{\theta}_k$ is a conditionally independent variable from the Gaussian distribution $\mathcal{N}(\breve{\theta}_k; \mu_k, \sigma_k^2)$.

ABI contains several essential integration steps that make the algorithm computationally fast and efficient. These integration steps rely, however, on the properties of the

Gaussian distribution. This is the main reason why ABI uses the probit link, since it restricts $P(\breve{\theta})$ to the Gaussian distribution.

Inference

In order to infer the parameter values, each corresponding auxiliary variable $\breve{\theta}_k \in \{\breve{\theta}_1, ..., \breve{\theta}_n\}$ needs to be estimated in terms of μ_k and σ_k^2. These variables can be estimated from the posterior distributions of $\breve{\theta}_k$, which is defined as:

$$P(\breve{\theta}_k \mid C_{1:M}, \mu_k, \sigma_k^2) \propto P(\breve{\theta}_k \mid \mu_k, \sigma_k^2) P(C_{1:M} \mid \breve{\theta}_k) \tag{2.27}$$

Here, $P(\breve{\theta}_l \mid \mu_l, \sigma_l^2) = \mathcal{N}(\breve{\theta}_l; \mu_l, \sigma_l^2)$ is the prior distribution of $\breve{\theta}_l$. Considering that the variable $\breve{\theta}_k$ is conditionally independent, its marginal likelihood function can be obtained by integrating out all $\breve{\theta}_l$'s except $\breve{\theta}_k$ from the likelihood function:

$$P(C_{1:M}|\breve{\theta}_k) = \int_{\mathbb{R}^{n-1}} P(C_{1:M}|\breve{\theta}_1, \ldots, \breve{\theta}_n) \prod_{l \neq k} (P(\breve{\theta}_l|\mu_l, \sigma_l^2) \mathrm{d}\breve{\theta}_l) \tag{2.28}$$

To prepare for the subsequent inference steps, the ABI method proposes to expand the likelihood equation (2.26) as the following form:

$$P(C_{1:M}|\breve{\theta}_1, \ldots, \breve{\theta}_n) = \sum_{t=1}^{T} \left(\alpha_t \prod_{l=1}^{n} \Phi^{o_{tl}}(\breve{\theta}_l) \right) \tag{2.29}$$

where the power of $\Phi(\breve{\theta}_l)$ in the t-th term of the expansion is indicated by o_{tl}. Therefore, by substituting (2.29) in (2.28) the integral can be written as:

$$P(C_{1:M}|\breve{\theta}_k) = \sum_{t=1}^{T} \left(\alpha_t \Phi^{o_{tk}}(\breve{\theta}_k) \prod_{l \neq k} c_{lo_{tl}} \right), \tag{2.30}$$

for all variables $\breve{\theta}_k$ and all powers $0 \leq o \leq O_k$, where O_k is the order of $\Phi(\breve{\theta}_k)$. The definition of c_k is given in (2.32). Now, (2.30) can be defined in the standard form:

$$P(C_{1:M}|\breve{\theta}_k) = a_{O_k} \Phi^{O_k}(\breve{\theta}_k) + \cdots + a_1 \Phi(\breve{\theta}_1) + a_0 \tag{2.31}$$

It is worth noting that the computation of vector $\mathbf{a}$ is the only click model dependent computation for this inference method, meaning that the remainder of this general inference algorithm can be adopted for every click model.

As a result of the marginal likelihood function (2.31), the posterior distribution of variable $\breve{\theta}_k$ (2.27) is no longer Gaussian, which makes the inference method inefficient. For this reason, the ABI method uses an online learning scheme called Gaussian density filtering [5]. This method first approximates the posterior distribution $P(\breve{\theta}_k \mid C_{1:M}, \mu_k, \sigma_k^2)$ as a Gaussian distribution $q(\breve{\theta}_k|\hat{\mu}_k, \hat{\sigma}_k^2) = \mathcal{N}(\breve{\theta}_k, \hat{\mu}_k, \hat{\sigma}_k^2)$. This approximation is acquired by the minimisation of the Kullback-Leiber (KL) divergence between $P(\breve{\theta}_k \mid C_{1:M}, \mu_k, \sigma_k^2)$ and $q(\breve{\theta}_k|\hat{\mu}_k, \hat{\sigma}_k^2)$, which can be reduced to the computation of the following integrals:

$$c_{ko} = \int_{-\infty}^{\infty} \Phi^o(\breve{\theta}_k) P(\breve{\theta}_k|\mu_k\sigma_k^2)\mathrm{d}\breve{\theta}_k \tag{2.32}$$

$$u_{ko} = \int_{-\infty}^{\infty} x\Phi^o(\breve{\theta}_k) P(\breve{\theta}_k|\mu_k\sigma_k^2)\mathrm{d}\breve{\theta}_k \tag{2.33}$$

$$v_{ko} = \int_{-\infty}^{\infty} x^2\Phi^o(\breve{\theta}_k) P(\breve{\theta}_k|\mu_k\sigma_k^2)\mathrm{d}\breve{\theta}_k \tag{2.34}$$

A detailed approach to compute these integrals efficiently can be found in [5].

When $\mathbf{a}$, c_k, u_k and v_k are computed, the posterior distribution of $\breve{\theta}_k$ can be approximated in terms of $\hat{\mu}_k$ and $\hat{\sigma}_k^2$:

$$\hat{\mu}_k = \frac{\sum_{k=0}^{O_k} a_o u_{ko}}{\sum_{o=0}^{O_k} a_o c_{ko}}, \tag{2.35}$$

$$\hat{\sigma}_k^2 = \frac{\sum_{o=0}^{O_k} a_o v_{ko}}{\sum_{o=0}^{O_k} a_o c_{ko}} - \hat{\mu}_k^2. \tag{2.36}$$

The updating procedure for a specific variable $\breve{\theta}_k$ is completed after the parameters μ_k and σ_k^2 are updated with $\hat{\mu}_k$ and $\hat{\sigma}_k^2$. Consequently, the approximation of the posterior is treated as the prior distribution for the next query session. This way, the ABI algorithm will update the variables incrementally as new query sessions are loaded into the click model.

Finally, once this algorithm is finished updating μ_k and σ_k^2, the value of each parameter θ_k can be estimated by the expectation of $\Phi(\breve{\theta}_k)$ with respect to $P(\breve{\theta}_k \mid \mu_k, \sigma_k^2)$:

$$\theta_k = \int_{-\infty}^{+\infty} \Phi(\breve{\theta}_k)\mathcal{N}(\breve{\theta}_k; \mu_k, \sigma_k^2)\mathrm{d}\breve{\theta}_k = \Phi\left(\frac{\mu_k}{\sqrt{\sigma_k^2 + 1}}\right) \tag{2.37}$$

Chapter 3

Methodology

Having outlined the theoretical foundations of this research, this section will elaborate on the approaches and methods that were used in the course of this thesis.

3.1 Approach

For this study, a click model was implemented that complements click data with human judged data in order to characterise noise in clicks. While adding the ability to characterise noise, this click model upholds the assumptions of the UBM click model. The human judged data was extracted by a search engine and includes binary relevance judgements together with their corresponding query-url pair. The click model is able to classify all clicks on judged urls in two categories — noise-free and noisy clicks — by assuming that this human judged data is the absolute truth. More specifically, all clicks on urls which are labelled as relevant can be defined as a noise-free click, while clicks on irrelevant labelled urls can be considered as a noisy click. It is important to characterise the latter, since these noisy clicks might not provide a good indication of relevance. If all clicked urls were human labelled, the noise of each click could be easily identified. However, since this would be very labour intensive, the amount of human judged data is limited. It is however possible to collect a large amount of click data at a very low cost. This model makes an attempt to generalise human labelled data to characterise the noise of each click, with the goal to estimate the relevance indication for unlabelled urls more accurately.

In order to generalise the labelled data, this study provides the contextual informa-

Feature Name	Feature Description
FirstQuery	True if it is the first query in the search session
TimeToLastAction	Time to last query or click

Table 3.1: Features used to predict noise

tion by using different features, which are listed in Table 3.1. It seems likely that a large amount of the first queries are navigational queries in which users are looking for a specific website. Subsequent queries, however, tend to be obscure or not well formulated and can thus lead to noisy clicks. This motivates the choice to take 'FirstQuery' as the first feature to characterise noise. It can be argued that the elapsed time since the last click or query (in the same search session) resembles the user's satisfaction with the search results. A satisfied user might therefore show other click behaviour than an unsatisfied user. Conclusively, the feature 'TimeToLastAction' is used to differentiate click behaviour in order to characterise noise.

The click model was designed to employ the human judged data in an attempt to develop a predictor that can characterise the relationship between values of the contextual features and the degree of noise. With this trained predictor, the click model is able to predict noise of a large amount of clicks without human judged data (by exploiting the context information in which a user makes a click). The next section will provide a detailed description of the training and the inference process.

Furthermore, an attempt has been made to analyse the performance of the click model under different sizes of human judged datasets. Based on this analysis, this study aims to gain a better understanding with regards to what extent the amount of human judged urls contribute to the performance of the noise-aware click model.

Finally, this study analysed the usefulness of the applied contextual features to characterise noise in order to improve the accuracy of click prediction.

3.2 Inference Method

In order to estimate the parameters of the N-UBM click model, the model is given click data and a human judged dataset which contains queries whose corresponding urls have

been judged. The labelled urls are used to train a noise predictor that can characterise the noise degree of a click. To estimate the noise for unlabelled urls, the noise predictor uses the weights w_j $(j = 1, ..., n)$ to estimate the degree of noise from the contextual feature values f_j $(j = 1, ..., n)$ of the click. For a given feature vector $\mathbf{f}$, the noise predictor estimates the probability of noise by $\Phi(\mathbf{w}^T\mathbf{f})$. Naturally, the prediction of noise becomes more accurate as more human judged urls and their relationship to the corresponding feature values are observed. This section will give a detailed illustration of the implementation of the inference procedure.

3.2.1 Approximate Bayesian Inference for N-UBM

Given the noise-aware UBM click model assumptions described in Chapter 2.2.2, the following parameters are defined for this model: $\mathbf{w}$ and $\mathbf{\Theta}$. Here, $\mathbf{w} = (w_1, ..., w_n)$ are the coefficients of the noise predictor and $\mathbf{\Theta} = (\alpha, \gamma, b)$ are probability parameters defined in the assumptions of N-UBM, where α is the relevance parameter, γ is the examination parameter and b is the parameter that measures the click probability in a noisy context.

To estimate $\mathbf{w}$ and $\mathbf{\Theta}$, this model employs the Approximate Bayesian Inference algorithm described in Chapter 2.3. This is an incremental updating algorithm, which means that query sessions are sequentially loaded into this model and the data is discarded after processing. When a new query session s comes in, the probability distribution of each parameter is updated. Before the parameters are updated, both $\mathbf{w}$ and $\mathbf{\Theta}$ have a prior distribution, namely $P(\mathbf{w})$ and $P(\mathbf{\Theta})$. The computed likelihood function $P(s \mid w_j)$ and $P(s \mid \theta_k)$ are then multiplied with the prior distribution in order to derive the posterior distribution $P(w_j \mid s)$ and $P(\theta_k \mid s)$. The algorithm concludes an update step by using the posterior as prior in the processing of the next query session.

As shown in the theoretical framework (Chapter 2.3.2), the ingredients needed to apply the ABI algorithm for updating the parameters are the session data and the likelihood function. Therefore, the likelihood function of the N-UBM model has to be derived from its specifications, so that the general inference algorithm can be adopted. In the inference process, queries and urls are denoted by q_i and u_j, where i and j indicate the index of a specific query or url. Thereby, it is assumed that q_i is the current query session and $\phi(j)$ indicates the index of the url at position j. The binary variable L_j

indicates if the j-th url is human-rated as relevant or not. Furthermore, the amount of url impressions in a query session are denoted by M.

The first step to derive the likelihood function of the N-UBM model is to define the parameters via probit links. Now, all parameters θ_k are connected with a new variable $\breve{\theta}_k \in (-\infty, +\infty)$ that is Gaussian distributed via the cumulative distribution function of the standard normal distribution (probit link). As such, $\alpha_{i\phi(j)} = \Phi(\breve{\alpha}_{i\phi(j)})$, $\gamma_{rd} = \Phi(\breve{\gamma}_{rd})$ and $b = \Phi(\breve{b})$. The updating of each w_j parameter, however, proves a more difficult process than the updating of each parameter θ_k, since w_j is a real number instead of a probability which is not applicable in the ABI algorithm. To solve this problem, an auxiliary variable $y = \mathbf{w}^T\mathbf{f}$ is introduced that represents the noise probability. The next section (3.2.2) will discuss how each $P(w_j \mid s)$ can be derived by using this variable.

and finally $y = \Phi(y)$. Subsequently, this study derived the following likelihood function from the N-UBM specifications:

$$
\begin{aligned}
&P(C_{1:M} \mid L_j, \breve{\alpha}, \breve{\gamma}, \breve{b}, y) = \\
&\prod_{j=1}^{M}(((\Phi(y)\Phi(\breve{b}) + (1-\Phi(y))\Phi(\breve{\alpha}_{i\phi(j)}))\Phi(\breve{\gamma}_{jd_j}))^{L}(\Phi(y)\Phi(\breve{b})\Phi(\breve{\gamma}_{jd_j}))^{1-L})^{C} \\
&\cdot((1-(\Phi(y)\Phi(\breve{b}) + (1-\Phi(y)))\Phi(\breve{\alpha}_{i\phi(j)}))\Phi(\breve{\gamma}_{jd_j}))^{L}(1\Phi(y))\Phi(\breve{b})\Phi(\breve{\gamma}_{jd_j}))^{1-L})^{1-C}
\end{aligned}
\tag{3.1}
$$

Here, d_j indicates the distance between position j and the last clicked url. In case there is no last click before j, it is assumed that $d_j = j$.

In case the click data is unlabeled with L_j unknown, the likelihood function is defined as:

$$
P(C_{1:M} \mid \breve{\alpha}, \breve{\gamma}, \breve{b}, y) = \prod_{j=1}^{M}((1-\Phi(y))\Phi(\breve{\alpha}_{i\phi(j)}) + \Phi(y)\Phi(\breve{b}))\Phi(\breve{\gamma}_{jd_j}) \tag{3.2}
$$

In line with the ABI method, the next step is to derive the $\mathbf{a}$ vector from the above likelihood functions (3.1) and (3.2). Given this polynomial function, the orders of $\Phi(\breve{\alpha})$, $\Phi(\breve{\gamma})$ and $\Phi(\breve{b})$ are always 1, i.e. $O_k = 1$ for every query session. Consequently, $\mathbf{a} = \{a_0, a_1\}$, which can be substituted in (2.31) to obtain the following marginal likelihood function of a specific variable $\breve{\theta}_k$:

$$
P(C_{1:M} \mid \breve{\theta}_k) = a_1\Phi(\breve{\theta}_k) + a_0 \tag{3.3}
$$

In order to determine the values of $\mathbf{a}$ for the corresponding variable $\breve{\theta}_k$ with regard to the u-th url in the query session, a variable ϵ is introduced that takes the product of the

likelihood function by considering all variables apart from $x_{i_{\phi(u)}}$ to be known.

$$\begin{aligned}\epsilon_u = \prod_{j \neq u}^{M} (((&\Phi(y)\Phi(\breve{b}) + (1 - \Phi(y))\Phi(\breve{\alpha}_{i\phi(j)}))\Phi(\breve{\gamma}_{jd_j}))^L \\ &\cdot (\Phi(y)\Phi(\breve{b})\Phi(\breve{\gamma}_{jd_j}))^{1-L})^C \\ &\cdot ((1 - (\Phi(y)\Phi(\breve{b}) + (1 - \Phi(y))\Phi(\breve{\alpha}_{i\phi(j)}))\Phi(\breve{\gamma}_{jd_j}))^L \\ &\cdot (1 - \Phi(y)\Phi(\breve{b})\Phi(\breve{\gamma}_{jd_j}))^{1-L})^{1-C}\end{aligned} \tag{3.4}$$

Given this variable ϵ, the $\mathbf{a}$ vectors for the corresponding variables are computed as follows.

For $\Phi(\breve{\alpha})$, the values for $\mathbf{a}$ are:

$$\mathbf{a} = \begin{cases} a_0 = \Phi(\breve{\gamma}_{ud_u})\Phi(\breve{b})\Phi(y)\epsilon_u & \text{if } C_j = 1, L_j = 1 \\ a_1 = \epsilon_u\Phi(\breve{\gamma}_{ud_u})(1 - \Phi(y)) & \\ & \\ a_0 = return & \text{if } C_j = 1, L_j = 0 \\ a_1 = return & \\ & \\ a_0 = \Phi(\breve{\gamma}_{ud_u})\Phi(\breve{b})\Phi(y)\epsilon_u & \text{if } C_j = 1, L_j = \text{unknown} \\ a_1 = \epsilon_u\Phi(\breve{\gamma}_{ud_u})(1 - \Phi(y)) & \\ & \\ a_0 = \epsilon_u - \epsilon_u\Phi(\breve{\gamma}_{ud_u})\Phi(\breve{b})\Phi(y)) & \text{if } C_j = 0, L_j = 1 \\ a_1 = \epsilon_u\Phi(\breve{\gamma}_{ud_u})(\Phi(y) - 1) & \\ & \\ a_0 = return & \text{if } C_j = 0, L_j = 0 \\ a_1 = return & \\ & \\ a_0 = \epsilon_u - \epsilon_u\Phi(\breve{\gamma}_{ud_u})\Phi(\breve{b})\Phi(y)) & \text{if } C_j = 0, L_j = \text{unknown} \\ a_1 = \epsilon_u\Phi(\breve{\gamma}_{ud_u})(\Phi(y) - 1) & \end{cases} \tag{3.5}$$

For $\Phi(\breve{\gamma})$, the values for $\mathbf{a}$ are:

$$\mathbf{a} = \begin{cases} a_0 = 0 & \text{if } C_j = 1, L_j = 1 \\ a_1 = \epsilon_u(\Phi(y)(\Phi(\breve{b}) - \Phi(\breve{\alpha}_{i\phi(u)})) + \Phi(\breve{\alpha}_{i\phi(u)})) & \\ & \\ a_0 = 0 & \text{if } C_j = 1, L_j = 0 \\ a_1 = \Phi(\breve{b})\Phi(y)\epsilon_u & \\ & \\ a_0 = 0 & \text{if } C_j = 1, L_j = \text{unknown} \\ a_1 = \epsilon_u(\Phi(y)(\Phi(\breve{b}) - \Phi(\breve{\alpha}_{i\phi(u)})) + \Phi(\breve{\alpha}_{i\phi(u)})) & \\ & \\ a_0 = \epsilon_u & \text{if } C_j = 0, L_j = 1 \\ a_1 = \epsilon_u(\Phi(y)(\Phi(\breve{\alpha}_{i\phi(u)}) - \Phi(\breve{b})) - \Phi(\breve{\alpha}_{i\phi(u)}))) & \\ & \\ a_0 = \epsilon_u & \text{if } C_j = 0, L_j = 0 \\ a_1 = -\epsilon_u\Phi(\breve{b})\Phi(y) & \\ & \\ a_0 = \epsilon_u & \text{if } C_j = 0, L_j = \text{unknown} \\ a_1 = \epsilon_u(\Phi(y)(\Phi(\breve{\alpha}_{i\phi(u)}) - \Phi(\breve{b})) - \Phi(\breve{\alpha}_{i\phi(u)}))) & \end{cases} \tag{3.6}$$

For $\Phi(\breve{b})$, the values for $\mathbf{a}$ are:

$$\mathbf{a} = \begin{cases} a_0 = \epsilon_u \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u})(1-\Phi(y)) & \text{if } C_j = 1, L_j = 1 \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = 0 & \text{if } C_j = 1, L_j = 0 \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = \epsilon_u \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u})(1-\Phi(y)) & \text{if } C_j = 1, L_j = \text{unknown} \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = \epsilon_u (1-\Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u})(1+\Phi(y))) & \text{if } C_j = 0, L_j = 1 \\ a_1 = -\epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = \epsilon_u & \text{if } C_j = 0, L_j = 0 \\ a_1 = -\epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = \epsilon_u (1-\Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u})(1+\Phi(y))) & \text{if } C_j = 0, L_j = \text{unknown} \\ a_1 = -\epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \end{cases} \tag{3.7}$$

For $\Phi(y)$, the values for $\mathbf{a}$ are:

$$\mathbf{a} = \begin{cases} a_0 = \epsilon_u \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u}) & \text{if } C_j = 1, L_j = 1 \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})(\Phi(\breve{b}) - \Phi(\breve{\alpha}_{i\phi(u)})) & \\ \\ a_0 = 0 & \text{if } C_j = 1, L_j = 0 \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(\breve{b}) & \\ \\ a_0 = \epsilon_u \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u}) & \text{if } C_j = 1, L_j = \text{unknown} \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})(\Phi(\breve{b}) - \Phi(\breve{\alpha}_{i\phi(u)})) & \\ \\ a_0 = \epsilon_u (1 - \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u}) & \text{if } C_j = 0, L_j = 1 \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})(\Phi(\breve{\alpha}_{i\phi(u)}) - \Phi(\breve{b})) & \\ \\ a_0 = \epsilon_u & \text{if } C_j = 0, L_j = 0 \\ a_1 = -\epsilon_u \Phi(\breve{\gamma}_{ud_u})\Phi(y) & \\ \\ a_0 = \epsilon_u (1 - \Phi(\breve{\alpha}_{i\phi(u)})\Phi(\breve{\gamma}_{ud_u}) & \text{if } C_j = 0, L_j = \text{unknown} \\ a_1 = \epsilon_u \Phi(\breve{\gamma}_{ud_u})(\Phi(\breve{\alpha}_{i\phi(u)}) - \Phi(\breve{b})) & \end{cases} \tag{3.8}$$

The remainder of the inference process proceeds as in the general inference algorithm (described in Chapter 2.3.2) in order to smoothly update the distribution of each parameter of Θ. The next section will illustrate how the update process of the weights is accomplished.

3.2.2 Sum-Product Message Passing for Noise-Aware UBM

As mentioned in the previous section, the updating of each w_i is made more difficult as these weights are real values which are not fit to use in the ABI algorithm. A solution to this problem is to use an auxiliary variable $y = \mathbf{w}^T\mathbf{f}$.

Each $P(w_i \mid s)$ can be computed by integrating over y and all other weights in $\mathbf{w}$, as

indicated by $\mathbf{w}^{\setminus i}$ in the following formula.

$$\begin{aligned} P(w_i \mid s) &\propto \int (\prod_{i=1}^{n} P(w_i)) P(y \mid \mathbf{w}) P(s \mid y) \mathrm{d}\mathbf{w}^{\setminus i} \mathrm{d}y; \\ P(s \mid y) &\propto \frac{P(y \mid s)}{P(y)} = \frac{P(y \mid s)}{\int (\prod_{i=1}^{n} P(w_j)) P(y \mid \mathbf{w}) \mathrm{d}\mathbf{w}} \end{aligned} \tag{3.9}$$

Chen et al. [3] derived this formula by applying the *sum-product message passing* algorithm on a factor graph. This study derived the following equations to update the distributions for each w_j:

$$\hat{\mu}_{pi} = \frac{(\mu_p - \sum_{j \neq i} f_j w_j)}{f} \tag{3.10}$$

$$\hat{\sigma}^2_{pi} = \frac{(\sigma_p^2 + \sum |f_j| \, \sigma_j^2)}{f} \tag{3.11}$$

$$\hat{\mu}_{w_i|s} = \frac{(\hat{\sigma}^2_{pi} \mu_{w_i} + \sigma^2_{w_i} \mu_{pi})}{\sigma^2_{w_i} + \hat{\sigma}^2_{pi}} \tag{3.12}$$

$$\hat{\sigma}^2_{w_i|s} = \frac{1}{\frac{1}{\sigma^2_{w_i}} + \frac{1}{\hat{\sigma}^2_{pi}}} \tag{3.13}$$

Finally, when the training is completed, each w_j will have a distribution $P(w_j) = \mathcal{N}(w_j, \mu_j, \sigma_j^2)$. The noise probability of a given arbitrary feature vector $\mathbf{f}$ can therefore be predicted as the expectation of $\Phi(\mathbf{w}^T \mathbf{f})$:

$$P(N = 1) = E(\Phi(\mathbf{w}^T \mathbf{f})) = \Phi(\frac{\sum_{i=1}^{n} \mu_j f_j}{\sqrt{\sum_{i=1}^{n} \sigma_j^2 f_j^2 + 1}}) \tag{3.14}$$

Chapter 4

Evaluation

This section will evaluate the performance of NCM by comparing it with a classical click model in web search, namely UBM. Given the fact that NCM is implemented as an extension of UBM, the Noise-Aware UBM model will be referred to as N-UBM. The experiments of both UBM and N-UBM are conducted by using the ABI algorithm to update their parameters. The performance of both click models is evaluated by the criteria *perplexity* and *log-likelihood*.

4.1 Experimental Setup

The search sessions that are used to train and evaluate both click models are collected from click logs of a large commercial search engine, namely Yandex[1]. The human judged dataset includes query-url pairs and corresponding relevance labels and region IDs. The relevance labels were assigned with binary values by human judges of Yandex: relevant (1) or irrelevant (0). In this work, the region IDs were not taken into account. In total, this study pre-processed approximately 400 000 search sessions for which all the query-url pairs have a relevance judgement. The statistics of the complete dataset are described in Table 4.1. The search sessions of the complete dataset are divided into a training and test set of roughly the same size. Furthermore, for each query session in the search session, the contextual information is extracted for each feature in Table 3.1. Additionally, Table 4.2 shows a subset of the data that is made in order to run several experiments with a varying amount of human judged urls.

[1] `https://www.yandex.ru`

Description	Frequency
Search Sessions in Train Set	199 430
Search Sessions in Test Set	203 182
Queries in Train Set	215 840
Queries in Test Set	223 113
Unique Queries in Train Set	116
Unique Queries in Test Set	169
Average Query Frequency Train Set	≈ 1 860
Average Query Frequency Test Set	≈ 1 319
Human Judged Unique Queries	116
Human Judged Ratings	2 158 400

Table 4.1: Statistics of complete dataset

This study used the PyClick2 [8] framework that includes the ABI inference algorithm, the UBM model and the necessary evaluation methods. The N-UBM model was implemented as an extension of the UBM model from this framework.

4.2 Evaluation Criteria

4.2.1 Log-Likelihood

Log-likelihood (LL) is an often used metric for comparing different click models [9]. After a click model is trained on a training set, it is tested on unseen test data. For a single url impression in each query session in the test data, the log-probability of a click event is computed, given clicks that were observed by the trained model. The LL of a dataset with an amount of query sessions is then calculated by taking the average LL on all individual url impressions. The higher the value of LL, the better the performance. The optimal value is 0.

4.2.2 Click Perplexity

The prediction accuracy is evaluated by click perplexity, which is a widely used metric to measure the quality of a click model [2, 3, 5, 6]. Perplexity measures how “suprised”

Description	Frequency
Search Sessions in Train Set	7 051
Search Sessions in Test Set	6 906
Queries in Train Set	7 715
Queries in Test Set	7 634
Unique Queries in Train Set	30
Unique Queries in Test Set	28
Average Query Frequency Train Set	≈ 256
Average Query Frequency Test Set	≈ 272

Table 4.2: Statistics of subset of the data

the model is when a click is observed. The smaller the value of perplexity, the better the prediction accuracy. The perplexity of a perfect model is 1, the perplexity of a simple baseline model that predicts each click with probability 0.5 equals 2. It is computed for binary click events at each position in a query session individually, rather than the whole click sequence as in the log-likelihood computation. For each position j, the perplexity p_j is calculated as:

$$p_j = 2^{-\frac{1}{N}\sum_{n=1}^{N}(C_j^n \log_2 q_j^n + (1-C_j^n)\log_2(1-q_j^n))} \tag{4.1}$$

where C_j^n denotes the binary click event of the j-th url in the n-th query session and q_j^n is the click probability derived from the click model on the j-th position in the n-th query session. The perplexity of an entire dataset is computed by taking the average over all positions.

4.3 Experimental Results

The performance measures over all query sessions in the test set are listed in Table 4.3. A t-test has been performed to validate the significance of improvement from the best performing click model with respect to the other click model under each performance measure. P-values of the t-test that are less than 0.01 are indicated with ▼.

As can be observed in the results, contrary to expectation, N-UBM scores lower than UBM in terms of perplexity and LL. In fact, the results show that the p-values of the

t-test are all less than 0.01, which confirms that the click prediction accuracy of N-UBM is significantly worse than that from UBM.

Click Model	Log-Likelihood	Perplexity
UBM	-0.1938	1.2579
N-UBM	-0.2126 ▼	1.3060 ▼

Table 4.3: Comparison experimental results between UBM and N-UBM for complete testset

An additional performance comparison of the N-UBM model has been conducted in order to study the correlation between the amount of human judged urls and the performance in terms of click prediction. This experiment is performed by training and testing the click model several times with an increasing amount of human relevance ratings. In order to acquire the results in a feasible time, the different trials are performed on a smaller set of the data (see Table 4.2). The results are reported in Table 4.4.

Against all expectations, the overall performance in click prediction remains the same or even deteriorates as the amount of human judged urls increases. The initial expectation was that more human judged urls would result in an improved noise predictor allowing the model to better interpret user click behaviour.Surprisingly, the results show that the click prediction is less accurate in comparison with UBM. However, after a thorough analysis of the data and the results, these outcomes appear to be less surprising.

The results analysis starts by investigating the clicked urls in relation to their human relevance labels. When taking a closer look at the distribution of relevance labels that were assigned to clicked urls in the complete dataset (Figure 4.1), we see that solely 11 percent of all clicks were considered to be irrelevant by human judges. In other words, only 11 percent of all clicks in this dataset are in fact noisy. In comparison, the data that was used in the work of Chen et al. [3] showed that more than 28 percent of the clicked urls were considered irrelevant. Conclusively, it is hard to capture the characteristics of noise in a large amount of click data with such a small percentage of noise.

Additionally, it turns out that 48 percent of the url impressions in the dataset were not clicked but were considered relevant. From these statistics, it appears that relevance labels and clicks often do not correlate with one another. Given this observation, it is

#Human Judged urls	Log-Likelihood	Perplexity
0	-0.2668	1.3628
1000	-0.2808	1.3883
2000	-0.2809	1.3883
10 000	-0.2808	1.3883
50 000	-0.2804	1.3879
75 000	-0.2792	1.3866

Table 4.4: Performance comparison N-UBM for different sizes of human judged data

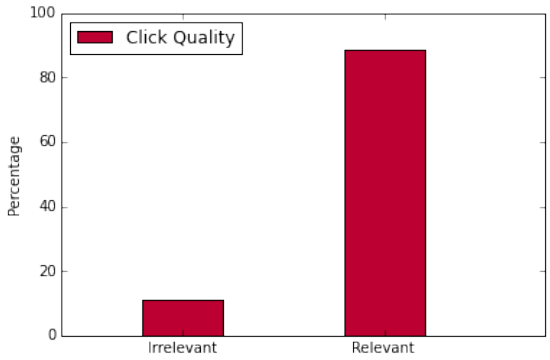

Figure 4.1: Distribution of Click Quality

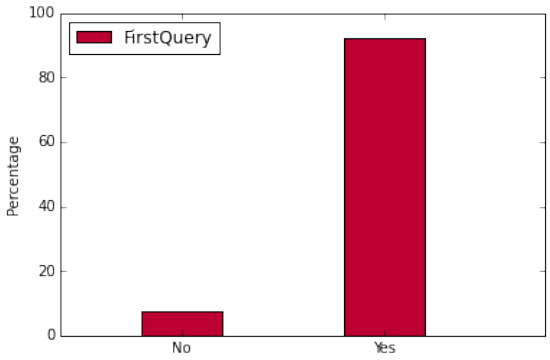

Figure 4.2: Frequency distribution of feature FirstQuery

highly likely that the general click behaviour under the queries in the dataset is mainly driven by rank. Possibly, the first urls that are presented in the SERP are clicked and satisfy the user's informational need, leaving other relevant-labelled urls unclicked. Since UBM is known for being successful in predicting rank-drive click behaviour, it appears that adding more assumptions and parameters to UBM only limits this model's capacities of predicting click behaviour [6].

NCM attempts to characterise the noise by learning the relation between human judgements (that determine the noise degree of a click) and values of contextual features. The features that are used in this experiment are 'FirstQuery' and 'TimeToLastAction'. The analysis of these features also may provide a possible explanation to the inferior

results of NCM with respect to UBM. Firstly, the frequency distribution of the feature 'FirstQuery' in Figure 4.2 shows that more than 92 percent of all url impressions occur in the first query session. Secondly, Figure 4.3 shows the frequency distribution of the feature 'TimeToLastAction' where more than 80 percent of all urls were shown on the SERP after an elapsed time of 0 seconds since the last query or click in the search session. The elapsed time of 0 zero seconds implies that these large amount of urls were shown in the first query of the search session (no action took place prior to this query). These two observations are explained by the fact that the vast majority of the search sessions consist of only one query session, which lead to skewed features values. Conclusively, these skewed feature values impede the task to learn a relation between these feature values and the degree of noise. Therefore, it can be argued that both features are not suited to characterise noise for this particular dataset.

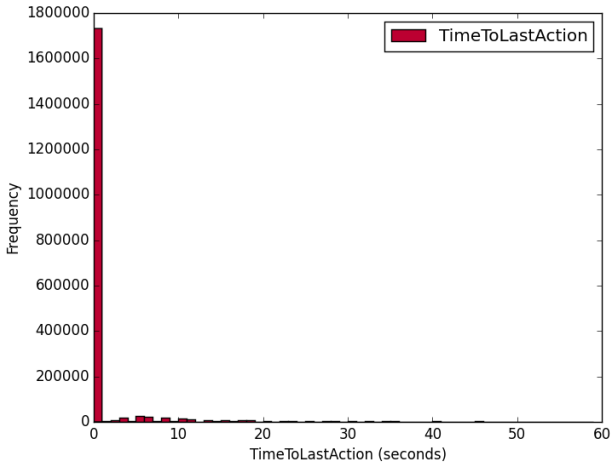

Figure 4.3: Frequency distribution of feature TimeToLastAction

Chapter 5

Conclusion

Click models can be used to infer the user-perceived relevance for each search result. As such, search engines can rank their search results according to the inferred relevance and list them on a SERP. Classical click models presume that every click is a good indication of relevance of the url in relation to the query. However, these models do not take into account that a high percentage of clicked results may be irrelevant to the user and will therefore give false indications of relevance. Thus, classical models have a considerable limitation by assuming every click as a binary event while neglecting the noise that may be involved.

In this work, a noise-aware click model has been implemented that characterises the noise degree in clicks. This is done by complementing click data with human relevance ratings to learn a noise predictor the relationship between contextual feature values and the degree of noise. As such, the model is also able to predict the noise for clicks which have not been rated by utilising the contextual information in which a click occurs. By taking the noise into account, the model can infer the estimated relevance of urls and predict the probability of click events. The particular noise-aware model upholds the assumptions of UBM and is therefore called N-UBM. In order to gain a better insight whether this model predicts the user's click behaviour accurately or not, this study compared the performance of N-UBM with the performance of UBM (in terms of click prediction). Additionally, the correlation between the performance of N-UBM and the amount of human judged ratings was studied. Finally, an analysis is made that describes the usefulness of the extracted contextual features in the interest of predicting

click behaviour.

The experimental results of this work show that, with the used click data and selected contextual features, N-UBM performs significantly worse with respect to UBM in terms of click prediction. Furthermore, it appears that the performance of N-UBM stays the same or deteriorates when more human judged ratings are added. From the used click data, two contextual features were extracted to characterise the noise that is associated with click events: 'FirstQuery' and 'TimeToLastAction'. From the statistical data analysis, it appears that both features are not suited to characterise the noise of this data as they produce very skewed values.

It seems, however, that the inferior performance of N-UBM can be attributed to the dataset that was used for this study. Firstly, statistical data analysis shows that only 11 percent of all clicks in this dataset are noisy. Hence, since not many clicked urls in the query sessions suffer from noise, the ability to capture noise is difficult. Secondly, 48 percent of the url impressions were not clicked despite the fact that they were considered relevant. These statistics indicate predominant rank-driven click behaviour under the queries in the dataset. Since UBM is known for being successful in predicting rank-driven click behaviour, it appears that adding more assumptions and parameters only limits its capacities to predict rank-driven click behaviour. To conclude, the statistics demonstrate that the majority of search sessions in the dataset contains only one query. As a consequence, the used features produce very skewed feature values, which makes it a very difficult task to learn the relation between these values and the degree of noise.

For further research it would be interesting to compose a different dataset with a higher percentage of noise in clicks, to evaluate if the model can capture this noise. Additionally, the dataset could contain more queries per search session, which would prevent skewed feature values. Finally, different contextual features could be extracted to analyse their effect on the performance of N-UBM. Hence, in order to accurately measure the effectiveness of N-UBM, the dataset of future research should satisfy these criteria and include additional features.

Bibliography

[1] Diane Kelly and Jaime Teevan. "Implicit feedback for inferring user preference: a bibliography". In: *ACM SIGIR Forum*. Vol. 37. 2. ACM. 2003, pp. 18–28.

[2] Hongning Wang et al. "Content-aware Click Modeling". In: *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2013, pp. 1365–1376.

[3] Weizhu Chen et al. "A Noise-aware Click Model for Web Search". In: *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM. 2012, pp. 313–322.

[4] Laura A Granka, Thorsten Joachims, and Geri Gay. "Eye-tracking analysis of user behavior in WWW search". In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2004, pp. 478–479.

[5] Yuchen Zhang et al. "Learning Click Models via Probit Bayesian Inference". In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 439–448.

[6] Georges E Dupret and Benjamin Piwowarski. "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*. ACM. 2008, pp. 331–338.

[7] Botao Hu, Nathan N Liu, and Weizhu Chen. "Learning from click model and latent factor model for relevance prediction challenge". In: *Proceedings of the Second Workshop on Web Search Click Data (WSCD)*. 2012.

[8] Aleksandr Chuklin et al. *PyClick2 - Click Models for Web Search*. June 2015. URL: `https://bitbucket.org/markil/pyclick2`.

[9] Fan Guo et al. "Click chain model in web search". In: *Proceedings of the 18th international conference on World wide web*. ACM. 2009, pp. 11–20.