

Vrije Universiteit Amsterdam

Universiteit van Amsterdam



Master Thesis

Position Bias Estimation for Unbiased Learning-to-Rank in Product Search

Author: Maarten Petit (12488194)

<i>1st supervisor</i>	Dr. Adam S.Z. Belloum	University of Amsterdam
<i>daily supervisor:</i>	Tom Steenbergen	Picnic Technologies B.V.
<i>2nd reader:</i>	Dr. Ana M. Oprescu	University of Amsterdam

*A thesis submitted in fulfillment of the requirements for
the joint UvA-VU Master of Science degree in Computer Science*

August 27, 2020

“All models are wrong, but some are useful”

George E.P. Box

Abstract

User interaction data provides an invaluable source of training data for learning-to-rank models. The cost-effectiveness of collecting user interaction data and the abundance of implicit feedback that can be obtained from it recently increased the interest in learning-to-rank from user interactions. However, a challenge in learning from user interaction data is the inherent bias and noise, most notably position bias. The unbiased learning-to-rank framework enables learning of unbiased models from biased and noisy user interaction data. The critical component in unbiased learning-to-rank is propensity estimation employed to re-weight user interactions using Inverse Propensity Scoring. In the unbiased learning-to-rank framework, propensity is equivalent to position bias. We propose a novel system for position bias estimation for an online grocery store and similar e-commerce stores. The system employs the Position-Based Propensity Model utilizing add to basket events. The parameters of the model are learned through a scalable implementation of the Expectation-Maximization algorithm. The Position-Based Propensity Model significantly outperforms both baseline models, the Document-Based CTR and Rank-Based CTR model, in terms of the log-likelihood score on the observed test data. Furthermore, the novel noise filtering extension of the Position-Based Propensity Model, employing order data to overcome trust bias, significantly increases the performance of the model. We find that the Position-Based Propensity Model estimates a higher position bias compared to other e-commerce scenarios considered in the literature. Most likely due to the specific characteristics of an online grocery store and how customers interact with the application. We experiment with a novel extension of the model that utilizes display events but find that — based on our experiments and data — the proposed extension cannot outperform the standard Position-Based Propensity Model. Furthermore, we investigate the influence of personalized ranking on position bias and find that personalization of search rankings increases position bias in the user interaction data.

Preface

This thesis concludes the two year program of the joint Master Computer Science at the Vrije Universiteit Amsterdam and University of Amsterdam. The six-month Master project was carried out as a research internship at Picnic Technologies, in the Advanced Analytics and Algorithms team. The internship was a great opportunity to apply both my professional and academic skills to the full extent. I would like to start by thanking Picnic Technologies for the opportunity to join the company as an intern and providing such an engaging research project, even in these uncertain times of a global pandemic. I would like to express my sincere gratitude to my daily supervisor Tom Steenbergen for his invaluable guidance and feedback during the project. I would also like to thank Dr. Adam Belloum for his guidance on the completion of the Master project. Furthermore, I would like to thank the team members of the Advanced Analytics and Algorithms team for their enthusiasm, inspiration and feedback. I would also like to express my gratitude to Harrie Oosterhuis from the Information and Language Processing Systems (ILPS) group of the University of Amsterdam. His feedback and expert-knowledge on the topics of position bias estimation and unbiased learning-to-rank provided valuable insights for this thesis.

Finally, I want to thank my family and close friends for their encouragement and support throughout my studies.

Contents

List of Figures	v
List of Tables	vi
1 Introduction	1
2 Problem Description	4
2.1 Problem Definition	4
2.2 Research Questions	5
2.3 Data	6
3 Theoretical Background	11
3.1 Learning-to-Rank in Information Retrieval	11
3.2 User Interaction Data	12
3.3 Bayesian Networks	13
3.3.1 Parameter Learning	13
3.4 Click Models	14
3.4.1 Document-Based CTR Model	16
3.4.2 Rank-Based CTR Model	16
3.4.3 Position-Based Model	17
3.5 Unbiased Learning-to-Rank	18
4 Related Work	22
4.1 Result Randomization Methods	22
4.2 Historical Click Log Methods	24
4.2.1 Standard Expectation-Maximization	25
4.2.2 Regression Based Expectation-Maximization	25
4.2.3 Embedded in Discriminative Models	26
4.2.4 Intervention Harvesting	27

CONTENTS

4.2.5	Dual-Learning Algorithm	28
4.2.6	Simplified Likelihood	28
5	Methodology	30
5.1	Position Bias Estimation Method	30
5.2	Models	32
5.2.1	Baseline Models	32
5.2.2	Position-Based Propensity Model	33
5.3	Data Pre-Processing	35
5.4	Evaluation	37
5.4.1	Hypothesis Testing	38
5.5	Experiments	39
5.5.1	Expectation-Maximization Iteration Threshold	39
5.5.2	Expectation-Maximization Initial Values	40
5.5.3	Expectation-Maximization Data Smoothing	40
5.5.4	Noise Aware Position-Based Propensity Model	40
5.5.5	Display Events Position-Based Propensity Model	41
5.5.6	Personalized Ranking	42
6	Implementation	44
6.1	Position-Based Propensity Model	44
6.2	Production System	44
6.2.1	Scalability	47
6.3	Potential Use Cases	48
7	Results	49
7.1	Sample Size	49
7.2	Expectation-Maximization Iteration Threshold	50
7.3	Model Performance	50
7.4	Position Bias Estimation	51
7.5	Experiments	54
7.5.1	Expectation-Maximization Initial Values	54
7.5.2	Expectation-Maximization Data Smoothing	54
7.5.3	Noise Aware Position-Based Propensity Model	55
7.5.4	Display Events Position-Based Propensity Model	55
7.5.5	Personalized Ranking	56

CONTENTS

7.6 Scalability	57
8 Conclusion	60
8.1 Future Work	60
8.2 Recommendations	61
References	63
Appendix	70

List of Figures

2.2	Query-rank frequency distribution log-log plot.	9
2.3	Query rank-frequency distribution (truncated at rank 5000 of total 309,260). 10	
3.1	Graphical representation of the Position-Based Model.	17
5.1	Window size experiment <i>expanding window</i> strategy.	38
5.2	Time ordered k-fold <i>sliding window</i> strategy.	39
6.1	Overview of the production system.	46
6.2	Overview of output tables in the Picnic data warehouse.	46
7.1	Model performance with respect to sample size (in day buckets).	50
7.2	PBM EM convergence plot truncated at iteration 20.	50
7.3	Average model performance over all folds, error bars indicate 95% CI.	51
7.4	Average model performance over all folds on non-personalized data, error bars indicate 95% CI.	51
7.5	Position bias estimated by the standard Position-Based Propensity Model.	54
7.6	Position bias estimated by the noise aware model extension.	56
7.7	Position bias estimated by the display events model extension.	56
7.8	Estimated position bias for the filter based personalization experiment.	58
7.9	Estimated position bias for both personalization experiments.	58
8.1	Position bias estimated by the standard Position-Based Propensity Mode for all positions.	70
8.2	Convergence plot initial values experiment $\{\theta_k\}, \{\gamma_{q,d}\} = 0.2$	74
8.3	Convergence plot initial values experiment $\{\theta_k\}, \{\gamma_{q,d}\} = 0.5$	74
8.4	Convergence plot initial values experiment 5.10.	74
8.5	Convergence plot initial values experiment 5.11.	74

List of Tables

2.1	General statistics on the Picnic in-app search engine.	7
2.2	Frequency of events captured on the in-app search page of Picnic.	8
2.3	Distribution of the number of add to basket events per search session.	9
3.1	Table of notation used in thesis.	21
7.1	Model performance differences, * indicates statistically significant.	51
7.2	Log-likelihood score of implemented models over all folds.	52
7.3	Model performance differences of proposed extension, * indicates statistically significant.	55
7.4	Log-likelihood score of PBM display events extension on fold 13.	56
7.5	Data size statistics and PBM training time (in seconds) over all folds.	59

Chapter 1

Introduction

In the intersection of information retrieval and machine learning an increasing number of systems and algorithms learn from user interaction data [1]. The cost-effectiveness of collecting user interaction data and the abundance of implicit feedback that can be obtained from it recently increased the interest in models that learn from user interactions for the task of ranking in search and recommendation [2, 3, 4]. Specifically in the context of search result ranking — which is the focus of this thesis — learning from user interactions has several advantages over traditional learning methods. Traditionally, most learning-to-rank systems are trained on annotated datasets collected from expert annotations or crowdsourcing. The annotated datasets are both costly to collect and inherently stationary, hence they can become obsolete quickly [4, 5, 6]. Furthermore, implicit feedback reflects the time-varying user preferences more closely compared to annotated datasets [7]. However, a challenge in learning from user interactions is the inherent bias and noise. The problem is that user interactions are not an absolute relevance signal but are affected by various bias factors such as selection bias [2], presentation bias [8], quality-of-context bias [7], trust bias [9, 10] and most notably position bias [2, 9, 11, 12, 13, 14]. Among these biases, position bias has the strongest influence on user interactions [3]. Thus, the implicit feedback obtained from the user interactions is inherently biased and potentially noisy. Consequently, naive use would lead to biased and less effective ranking models [2, 3]. To leverage the full potential of implicit feedback obtained from user interaction data, models that learn from user interactions should account for bias. Recent work introduced the unbiased learning-to-rank framework, which employs a counterfactual inference approach that is proven to learn an unbiased ranking model from biased user interaction data [2, 3]. In this framework the click propensity estimation — which is equivalent to position bias estimation under the unbiased learning-to-rank framework — and the learning-to-rank

1. INTRODUCTION

problems are separated. The critical component in this framework is to estimate position bias. In this thesis we address the problem of position bias estimation in the unbiased learning-to-rank framework for an online grocery store, namely Picnic.

Picnic is Europe’s fastest growing online grocery store that aims at making online grocery shopping affordable and convenient. Picnic is an app-only online grocery store currently operating in The Netherlands and Germany that delivers the groceries directly to the customer without operating any brick-and-mortar stores. The mobile application — in that sense — is the store of Picnic. Consequently, the convergence of an app-only storefront and the aim to make grocery shopping convenient makes it is essential to provide customers with a seamless in-app experience. Picnic tracks in-app behavior of customers and collects this information using an app event data pipeline. The resulting click stream data includes information on practically any user interaction of a customer in the Picnic app. This user interaction data is crucial for improving customer experience.

It is essential in any store that the customer can find the desired products and the same goes for the online store of an online grocery like Picnic or any other e-commerce store. Search and ranking fulfill an important role in this user journey. The in-app search functionality and the ranking involved are crucial to ensure user satisfaction by presenting the most relevant products to the user. Furthermore, optimizing the search ranking can also be used to maximize business metrics such as product conversion and revenue [15]. Learning-to-rank for e-commerce search is an emerging new application of machine learning in information retrieval. To effectively train a learning-to-rank model a considerable amount of labeled training data is required. Learning-to-rank models trained on user interaction data can be employed for this purpose [5, 15]. However, the user interaction data of Picnic is also affected by the aforementioned biases.

In this thesis we focus on position bias estimation for unbiased learning-to-rank under the Position-Based Propensity Model. We propose and implement a system for position bias estimation for an in-app product search engine of an online grocery store — or a similar e-commerce scenario — employing a combination of historical click event data, customer order data and product display event data. Furthermore, we implement a production ready system for position bias estimation including a Python implementation of the Position-Based Propensity Model [6, 16, 17] optimized using the Expectation-Maximization algorithm [17, 18]. To the best of our knowledge, we are the first to estimate position bias in the unbiased learning-to-rank framework for an online grocery store or a similar e-commerce scenario, employing both add to basket click events and customer order data. Furthermore, we are the first to experiment with an extension of the Position-Based Propensity

1. INTRODUCTION

Model that utilizes product display events. Moreover, we are the first to estimate the effect of personalized ranking on position bias. The proposed system can be employed in any similar e-commerce scenario with a comparable number of products and queries.

The remainder of this thesis is organized as follows. In Chapter 2 we outline the context and problem definition, define the research questions and discuss the considered data. We outline the relevant theoretical background in Chapter 3. The related work on position bias estimation is discussed in Chapter 4. In Chapter 5 we discuss the proposed methodology of the thesis. The implementation details on the position bias estimation system are discussed in Chapter 6. In Chapter 7 the results of the thesis are discussed. Finally, the thesis is concluded in Chapter 8.

Chapter 2

Problem Description

In this chapter the problem of the thesis is formulated. First, we discuss the problem definition. Next, we define the main research question as well as the sub-questions. Last, a general outline of the Picnic data considered in the thesis is described.

2.1 Problem Definition

In-app product search ranking is an important part of an e-commerce store and learning-to-rank can be employed to improve this. User interaction data provides an abundant source of implicit feedback for such learning-to-rank models. However, this data is inherently biased and noisy. The unbiased learning-to-rank framework enables learning of unbiased ranking models even from biased implicit feedback. The critical component in this framework is position bias estimation. We consider the problem of position bias estimation under the unbiased learning-to-rank framework for an in-app product search engine of the online grocery store Picnic.

In recent research the problem of position bias estimation has been considered for web search [6], personal search [2, 13], hotel search [19] and e-commerce search [20]. Though, online grocery stores are a type of e-commerce, they differ in multiple aspects from the type of e-commerce considered by Aslanyan and Porwal [20]. First, an online grocery store comprises a significantly different and smaller assortment. An online grocery store and comparable e-commerce stores have a limited number of products (thousands to tens of thousands) compared to the type of e-commerce stores considered by Aslanyan and Porwal [20] such as eBay and Amazon where the number of products is at least a hundred fold (millions). Furthermore, customers interact differently with an online grocery store compared to other e-commerce stores. In an online grocery store, customers are familiar

2. PROBLEM DESCRIPTION

with the assortment, frequently order the same products and in many cases can directly add a product to the basket without the need to visit a detailed information page of the product.

The literature on position bias estimation for unbiased learning-to-rank includes no prior work on position bias estimation for an online grocery store or similar e-commerce stores. It is an open question how previously proposed methods for position bias estimation can be effectively applied and adapted to fully leverage the available data and characteristics of online grocery stores. Furthermore, there is no work carried out on how these methods can be implemented in an efficient and scalable system. Moreover, the effect of personalized ranking – which is common in online grocery stores – on position bias has not been studied. In the unbiased learning-to-rank framework no prior work investigated how display event data can be utilized for position bias estimation. From this point on, we will refer to Picnic and assume that the concepts and solutions can be extended to any similar e-commerce store that compares to the aforementioned scenario.

In addition to the in-app search engine, the Picnic app provides several other ways for a customer to find and discover products. The app includes a category tree page where products are categorized in multiple levels, a personalized page with products that the customer purchased before as well as specific sets of curated product ranking pages such as promotions. The effect of position bias can occur throughout the different product rankings in the Picnic app. However, the focus of this thesis is on quantifying the position bias effect for the in-app search engine.

2.2 Research Questions

The main research question considered in this thesis is:

How can we develop and implement a system to estimate position bias for unbiased learning-to-rank in the context of Picnic?

To answer the main research question the following sub-questions are considered:

RQ1: What are characteristics of the Picnic search engine and the corresponding user interaction data?

RQ2: Which methods for position bias estimation exist in the literature and how do they compare?

2. PROBLEM DESCRIPTION

***RQ3:** Which method for position bias estimation can we employ and adapt in the case of Picnic?*

***RQ4:** How can we improve the position bias estimation method utilizing specific characteristics and data of the Picnic in-app search engine?*

***RQ5:** How can we develop a production ready and scalable system for position bias estimation?*

***RQ6:** What are potential use cases of position bias estimation for Picnic?*

2.3 Data

In this section we consider the following sub-question:

***RQ1:** What are characteristics of the Picnic search engine and the corresponding user interaction data?*

The data considered for this research consists of user interaction data from the Picnic in-app search page. This page allows a customer u to enter a search term, referred to as query q and provides the user with a list of a variable number N of products D_q that match the given query. The products are presented in a $\frac{N}{2} \times 2$ grid of product tiles of which the four top most product tiles — or top most two rows — are always in the visible part of the screen. The customer can scroll down the list of results to bring the other products into the visible part of the screen. We consider the product tile positions in order from left to right and top to bottom. Thus, the upper left product tile is considered as position one and the upper right tile as position two, following this pattern down the result list. The product tile comprises of the price, contents or weight, product image as well as labels on product characteristics (e.g. bio). The products in the result set $D_q = \{d_1, \dots, d_N\}$ are ranked by a ranking function $f(q, d, u)$ that considers a combination of attributes on the query, product and customer. We will not discuss the details of the ranking system as this is proprietary information and not directly relevant for this thesis. However, we do emphasize the fact that the ranking function is personalized, meaning that based on the past purchases of a customer the result ranking is individually optimized for the respective customer. This ranking optimization induces that customers can be presented with divergent rankings for identical queries. The customer can interact with the search system in different ways. The different user interactions and corresponding frequencies, that are captured on the search

2. PROBLEM DESCRIPTION

Table 2.1: General statistics on the Picnic in-app search engine.

A large black rectangular box redacting the content of Table 2.1.

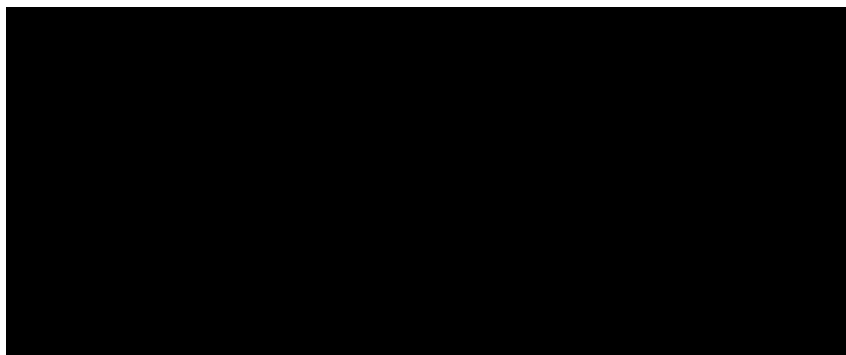
results page are listed in Table 2.2. First, the customer can directly interact with the product tiles in the search result list. The customer can click on a product tile to add a product to the basket. Subsequently, the customer can remove the item from the basket by interacting with the product tile. Furthermore, the display events indicate the interaction of scrolling in the result list and specifically that the product tile was in the visible part of the screen. Second, the customer can click on an information icon or long press a product tile to display a detailed information page on the product. The product detail page, containing detailed information on the product, comprises nutritional information, ingredients and information on the origin and supplier of the product. The customer can also add or remove products from the basket in the product detail page. Next to the events itself the system captures the context of every event. The context includes information on the user that initiated the search, the respective query, the product and the encompassing result list. The primary interest of this thesis is the event data in the search page but additionally related data on the context can be employed to improve the position bias estimation. The user interaction event dataset will be referred to as click log.

[REDACTED]

[REDACTED]

2. PROBLEM DESCRIPTION

Table 2.2: Frequency of events captured on the in-app search page of Picnic.



[REDACTED]

The rank-frequency distribution of the queries is depicted in Figure 2.3. We observe the long-tail distribution which is common in search engines [21]. A small quantity of queries has a high frequency whereas the largest quantity of queries has a low frequency, often occurring only once. The former group of queries are considered the head queries whereas

2. PROBLEM DESCRIPTION

Table 2.3: Distribution of the number of add to basket events per search session.

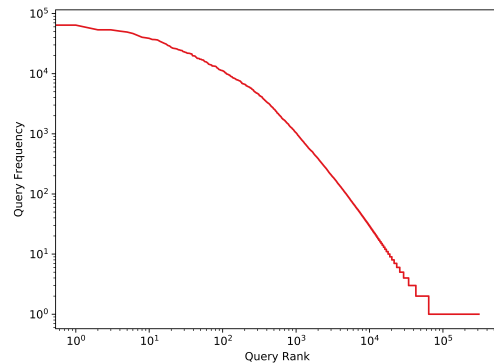
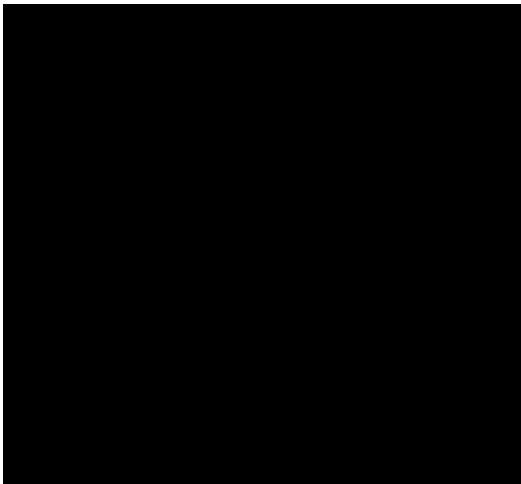
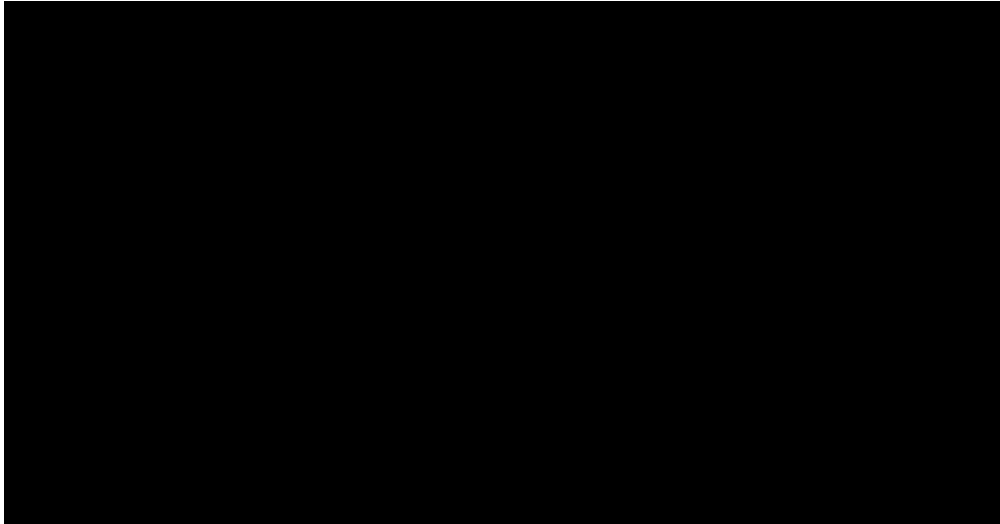


Figure 2.2: Query-rank frequency distribution log-log plot.

the latter group is considered the tail queries. Figure 2.2 depicts the log-log plot of the rank-frequency distribution. We observe that compared to web search, as described by Büttcher et al. [21], we have a larger quantity of head queries. This is revealed by the curve of the line, in web search the rank-frequency distribution shows a more straight line [21].

The dataset will be referred to as click log and is formalized for the remainder of this thesis as follows. We have click log S that consist of search sessions s containing a click log item (q, d, k, c) for every product included in the corresponding result list $(q, d, k, c) \in s \in S$. The click log items consist of q and d that denote the query and document identifiers respectively, $k = Z^+$ denotes the position on which the document was displayed and

2. PROBLEM DESCRIPTION

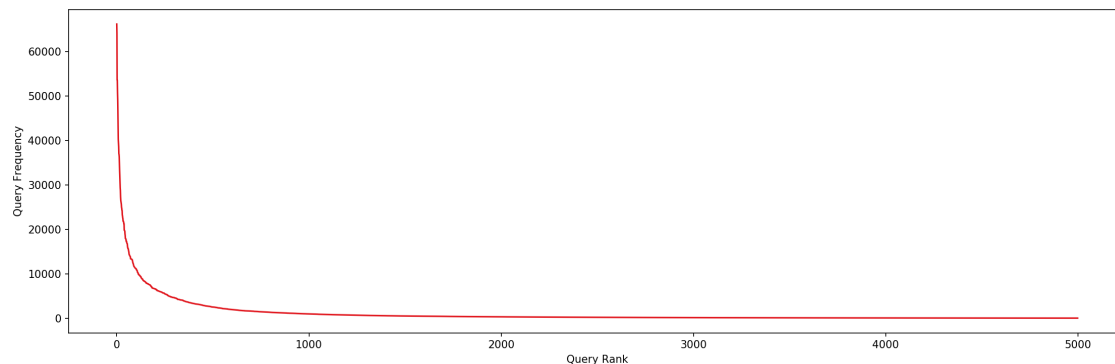


Figure 2.3: Query rank-frequency distribution (truncated at rank 5000 of total 309,260).

$c = \{1, 0\}$ denotes a click (add to basket event) or non-click on the document. In this thesis the query identifier is the corrected search term that the customer entered in the search bar. The search term is corrected by the production search system of Picnic. We could alternatively utilize the search term as typed by the user directly but this would yield a reduced number of events per search term as well as additional noise. The document identifier d is the product identifier as stored in the internal systems of Picnic. The product tile display events are denoted by v . Some more details will be discussed in Chapter 5.

Chapter 3

Theoretical Background

In this chapter we outline the theoretical foundation of this thesis and discuss the relevant theoretical background on various subjects. First, we will discuss the related elements of information retrieval, learning-to-rank and user interaction data. Next, we will discuss relevant theory on Bayesian Networks, including inference over parameters of such networks, which are employed for the models implemented in this thesis. Next, we will discuss the strongly related research on click models. Finally, we discuss the unbiased learning-to-rank framework and the essential role of position bias estimation in this framework.

3.1 Learning-to-Rank in Information Retrieval

Document retrieval is a task in information retrieval where the system maintains a collection of documents D and given a query q the system must retrieve a sub set of relevant documents D_q that match the query. Furthermore the system must rank this sub set of documents. For the task of ranking the documents $d \in D_q$ a ranking function is applied $f(q, d)$. Traditionally this ranking function is not trained but an increasing amount of machine learning models are employed for this task, specifically referred to as learning-to-rank models [22]. Learning to rank is a class of machine learning models that are utilized in information retrieval for the task of ranking documents, most notably in search result pages. Learning-to-rank algorithms can be classified based on the loss function used for learning into three categories, pointwise, pairwise and listwise methods [1, 23]. In the pointwise approach the model learns a numerical or ordinal score for each query-document pair individually. Subsequently, any set of document for a given query can be ordered or ranked using this relevance score. The pairwise approach is concerned with learning the relative order of two document for a given query. The listwise approach optimizes the

3. THEORETICAL BACKGROUND

ranking for a given query by taking the entire set of document as input. The pairwise and listwise algorithms usually outperform the pointwise algorithms [22]. We will not discuss details on the different approaches as we can assume that the position bias estimation can be employed in any of the learning-to-rank approaches considered in the unbiased learning-to-rank framework [3, 13, 24].

3.2 User Interaction Data

User interaction data presents a potential wealth of implicit feedback on user preferences. Thus, prior work has focused on extracting useful relevance signals from user interaction data. Studies on the decision process of the user on a search results page prove that clicks can be used as a relevance signal [7, 11]. However, these studies — that conduct several experiments and track the users decision process through eye-tracking — also prove that user interactions are not solely influenced by relevance but are affected by various biases. Thus, user interactions cannot be used as absolute relevance judgments. We describe the different types of biases by which user interaction data are affected and why we focus on position bias estimation for this thesis. First, it is proven that users examine the documents at higher ranked positions with a higher probability irrespective of the quality of the ranking. Especially after the first two ranks — that receive most attention — the examination probability drops fast. This effect and it’s influence on user interaction data is referred to as *position bias* [3, 9, 13]. Furthermore, user interactions are also influenced by the order of presentation. Users tend to click on the first result more often compared to the second result though both positions having similar examination frequencies. This effect on user interactions is caused by the trust of a user in the capability of a search ranking function to estimate the relevance of a document and is referred to as *trust bias* [9, 10, 11]. The overall quality of the ranking also affects user interactions and not solely that of the clicked document. This effect is referred to as *quality-of-context bias* [7, 9]. *Selection bias* is the effect that user interactions are limited to the presented documents in the result list [2]. The result attractiveness also has a significant influence on the click of a user, this effect is referred to as *presentation bias* [8, 14].

In this thesis we focus on position bias estimation for three reasons. First, position bias has the most affect on user interactions of all the aforementioned biases. Second, position bias can be effectively estimated through various different methods as proven in the literature. Finally, position bias can be leveraged in the unbiased learning-to-rank framework to learn a proven unbiased ranker.

3. THEORETICAL BACKGROUND

3.3 Bayesian Networks

Bayesian Networks, used for representation, inferences and learning, are probabilistic graphical models that represent the joint probability distribution of a set of random variables denoted $X = \{X_1, \dots, X_n\}$ [25]. In probability theory a random variable is a function over the set of outcomes of an experiment. In this thesis it is assumed that events and consequently random variables are binary-valued $X = \{1, 0\}$. The distribution over such a random variable is the Bernoulli distribution $X \sim \text{Bernoulli}(\theta)$ where θ is the parameter of the distribution.

A Bayesian Network is a directed acyclic graph consisting of nodes and edges representing random variables and the conditional dependencies between the variables respectively. Bayesian networks consider an efficient factorized representation of the joint probability distribution that leverages conditional independence of variables on their non-descendants. Event A and B are conditionally independent given C , denoted by $(A \perp\!\!\!\perp B)$, if $P(A, B|C) = P(A|C)P(B|C)$ holds. The joint probability of random variables $X = \{X_1, \dots, X_n\}$ is denoted by $P(X_1, \dots, X_n)$ and can be factorized using the chain rule of conditional probabilities as formalized in 3.1. For a Bayesian Network we can obtain this factorization through 3.2 where $Pa(X_i)$ is the set of parents of random variable X_i .

$$P(A_1, \dots, A_n) = P(A_1)P(A_2|A_1) \dots P(A_n|A_1, \dots, A_{n-1}) \quad (3.1)$$

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i|Pa(X_i)) \quad (3.2)$$

3.3.1 Parameter Learning

Parameter learning or parameter estimation is the process of learning the joint distribution of the network using training data \mathcal{D} . The objective of parameters learning is to find the parameters that maximize the likelihood of observing data $\mathcal{D} = \{x_1, x_2, \dots, x_n\}$. We can obtain the likelihood for parameter θ relative to \mathcal{D} using 3.3. Maximizing the log-likelihood 3.4 is equivalent to maximizing the likelihood as the two are monotonically related and due to the favorable computational properties the log-likelihood is used in practice. If the dataset \mathcal{D} contains fully observed instances the parameters be estimated using maximum likelihood estimation or Bayesian inference approaches. However, if the instances are not fully observed, either due to latent parameters in the network or incomplete data, one needs to apply other parameter learning methods most notably Expectation-Maximization

3. THEORETICAL BACKGROUND

[18]. The Expectation-Maximization algorithm provides an iterative solution to maximum likelihood estimation with latent variables.

$$\mathcal{L}(\theta : \mathcal{D}) = \prod_{x_i \in \mathcal{D}} p^{x_i} (1 - p)^{1 - x_i} \quad (3.3)$$

$$\mathcal{L}\mathcal{L}(\theta : \mathcal{D}) = \sum_{x_i \in \mathcal{D}}^n x_i (\log p) + (1 - x_i) \log(1 - p) \quad (3.4)$$

3.4 Click Models

Click Models are a family of models with the main purpose of modeling the behaviour of users on a Search Engine Result Page (SERP). The models are learned from user interaction data, generally click data, hence then name. Click Models are employed for click simulation experiments, evaluating search results and inferring document relevance. [17]

The intuitions behind different click models are based on hypothesis of user behaviour that are tested by means of experiments on real users. There are many types of user behaviour biases that different click models take into account by means of introducing random variables and dependencies between them.

Click models are often represented as Bayesian Networks but can also be represented as feature-based machine learning models. The conventional generative model approach employing Bayesian Networks aims to learn the joint probability of the observed and latent random variables in the network from historical click log data. The process of statistical inference in the context of click models, called parameter estimation, has two main techniques, namely maximum likelihood estimation (MLE) and Expectation-Maximization (EM). Two types of click models can be distinguished, models that contain only observed variables and models that have one or more unobserved variables. The parameters of models that solely involve observed variables can be estimated using MLE. Whereas parameters of models that include latent variables require estimation using the EM algorithm.

Click models range from simple models with a limited number of parameters to complex models that contain many parameters and relations. The most basic click model is the Random Click Model (RCM) [17] which assumes that every document has the same click probability. The simplistic RCM consist of a single parameter that can be estimated from observed click data and is often used as performance baseline. The RCM does not incorporate assumptions on the user behaviour. The basic assumptions on user behaviour are that the click of a user depends on the rank of document in the result list as well as the

3. THEORETICAL BACKGROUND

attractiveness or relevance of the document. These assumptions are incorporated in the Rank-Based CTR Model (RCTR) and the Document-Based CTR Model (DCTR) respectively [17]. These models are based on the concept of click-through rate (CTR) as formalized in 3.5. The CTR is the ratio of clicks and impressions where impressions is the number of times the document or position was included in the search result list [17]. The CTR models are simple models often used as performance baselines for more complex click models.

$$CTR = \frac{clicks}{impressions} \cdot 100\% \quad (3.5)$$

The Position-Based Model (PBM) [16, 26] consolidates the rank and relevance assumptions under the *examination hypothesis* as formalized in 3.6. The hypothesis states that a document is clicked if and only if the document is both examined E and perceived relevant R . The formal definition of the RCTR, DCTR and the PBM are given later in this section.

$$C_d = 1 \Leftrightarrow E_d = 1, R_d = 1 \quad (3.6)$$

The Cascade Model (CM) [16, 17] incorporates the additional assumption that a user sequentially examines the documents in the result list from top to bottom until the user finds a relevant document that is clicked, after which the search session is ended. More formally, the assumption is that the document at position k is examined if and only if the document at position $k - 1$ was examined and not perceived relevant. This is a stronger assumption on user behaviour compared to the Position-Based Model which does not assume sequential examination of the result list or that a sessions can only have a single click. More advanced click models are either an extension of the Position-Based Model, such as the User Browsing Model (UBM) [27] or the Cascade Model such as the Dependent Click Model (DCM) [28], Click Chain Model (CCM) [29] and the Dynamic Bayesian Network Model (DBN) [30].

The aforementioned click models are trained on click data exclusively. However, some click models aim to extend learning to other user interactions namely cursor movements and scrolling interactions [31, 32].

The concept of relevance R , as introduced in the different click models and the learning-to-rank framework is sometimes referred to as the perceived relevance or attractiveness. Relevance is considered to be a characteristic of the document as presented to the user in the search result list. The perceived relevance R is highly correlated with the true relevance \tilde{R} [6, 30, 33, 34]. However, there can be a substantial discrepancy between the perceived

3. THEORETICAL BACKGROUND

relevance R and actual relevance \tilde{R} . The degree of discrepancy depends on the scenario in which a model is employed. In the example of web search, the document — a web page — is presented in the search results list by means of a title and document snippet. The snippet reveals only a part of the full content of the web page that it directs to. Thus, the user can observe the full web page content after clicking and find that the web page is actually not relevant as proven by Chapelle and Zhang [30].

Click models that are used to infer document relevance scores can be considered learning-to-rank models of the pointwise category. However, the capabilities and performance are limited compared to recent learning-to-rank models in the pairwise and listwise category. A fundamental limitation of click models lies in the fact that the models can only infer parameters over query-document pairs that are included in past observations. Consequently, in the role of a learning-to-rank model, click models cannot learn the relevance of unknown query-document pairs. Furthermore, click models cannot infer reliable relevance estimations for tail queries as the models require queries to repeat multiple times. [3, 17]

3.4.1 Document-Based CTR Model

The Document-Based CTR Model [16, 17] incorporates the assumption that the click of a user depends on the relevance of document d for given query q . The model introduces a single parameter for each query-document pair as formalized in 3.7. The log-likelihood of observing given click log data S under the Document-Based CTR Model is obtained through 3.8.

$$\rho_{q,d} = P(C = 1|q, d) \quad (3.7)$$

$$\mathcal{LL}(S) = \frac{1}{|S|} \sum_{s \in S} \sum_{(c,q,d,k) \in s} c \cdot \log(\rho_{q,d}) + (1 - c) \cdot \log(\rho_{q,d}) \quad (3.8)$$

3.4.2 Rank-Based CTR Model

The Rank-Based CTR Model [17, 27] incorporates the assumption that the click of a user depends on the rank or position k of the document on the search result page. The model introduces a single parameter for each position in the search result page as formalized in 3.9. The log-likelihood of observing given click log data S under the Rank-Based CTR Model is obtained through 3.10.

$$\rho_k = P(C = 1|k) \quad (3.9)$$

3. THEORETICAL BACKGROUND

$$\mathcal{LL}(S) = \frac{1}{|S|} \sum_{s \in S} \sum_{(c,q,d,k) \in s} c \cdot \log(\rho_k) + (1 - c) \cdot \log(\rho_k) \quad (3.10)$$

3.4.3 Position-Based Model

The Position-Based Model incorporates the examination hypothesis 3.6. Thus, assuming that a click depends jointly on the examination E and relevance R 3.14. The model assumes that E and R are conditionally independent on C . The model incorporates the assumption that relevance $R_{q,d}$ depends on both the query q and document d by introducing a set of query-document dependent parameters $\gamma_{q,d}$ as formalized in 3.11. The probability of examination depends heavily on position of a result, to incorporate this assumption in the model a set of examination parameters θ_k as formalized 3.11.

$$P(E = 1|k) = \theta_k \quad (3.11)$$

$$P(R = 1|q, d) = \gamma_{q,d} \quad (3.12)$$

In this model the random variable $C_{q,d,k}$ is the only observed variable and both E_k and $R_{q,d}$ are unobserved latent parameters. A graphical representation of the model is depicted in Figure 3.1. In the model the only observed random variable $C_{q,d,k}$ has two parent nodes. As per 3.2, we can obtain the factorized representation of the model by taking the product of the probability of the parents E_k and $R_{q,d}$ of $C_{q,d,k}$ as formalized in 3.13. The model assumes that the first position is always examined as formalized in 3.15.

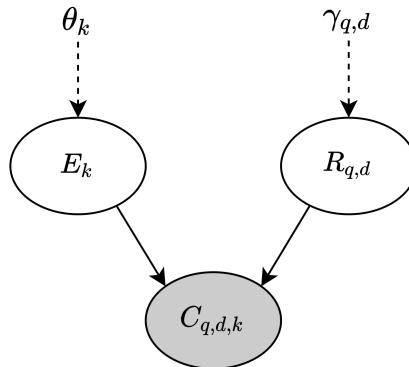


Figure 3.1: Graphical representation of the Position-Based Model.

3. THEORETICAL BACKGROUND

$$P(C = 1|q, d, k) = P(E = 1|k) \cdot P(R = 1|q, d) \quad (3.13)$$

$$C_d = 1 \Leftrightarrow E_k = 1, R_{q,d} = 1 \quad (3.14)$$

$$P(E = 1|k = 1) = \theta_1 = 1 \quad (3.15)$$

The log-likelihood of observing given click log data S under the Position-Based Model is obtained through 3.10.

$$\mathcal{LL}(S) = \frac{1}{|S|} \sum_{s \in S} \sum_{(c,q,d,k) \in s} c \cdot \log(\theta_k \gamma_{q,d}) + (1 - c) \cdot \log(1 - \theta_k \gamma_{q,d}) \quad (3.16)$$

3.5 Unbiased Learning-to-Rank

Traditional click models aim to learn the query-document relevance directly by taking position bias into account. The focus of these models is not on position bias estimation but rather on the modeling of user behaviour, click simulations and query-document relevance inference. Recent work on the unbiased learning-to-rank framework leverages position bias estimation to effectively learn unbiased learning-to-rank models from biased user interaction data. The unbiased learning-to-rank framework introduced by Joachims et al. [3] separates the click propensity estimation from the learning-to-rank algorithms. The framework enables the learning of sophisticated ranking models from implicit feedback employing the (separately) estimated propensities. In click models the user modeling and relevance learning are tightly coupled which limits the potential complexity and performance of the models. The unbiased learning-to-rank framework adopts the approach of inverse propensity scoring (IPS) from causal inference studies. This counterfactual inference framework allows unbiased learning even with biased implicit feedback.

Unbiased learning-to-rank [2, 3] generalized in Counterfactual Learning-to-Rank [35] aims to learn an unbiased ranking model from biased and noisy historical user interaction data. Traditional learning-to-rank methods assume that the relevance of all documents is known. This is referred to as full information feedback. However, due to bias and noise in user interaction data, we only observe relevance signals on the documents that the user interacted with. This is referred to as partial information feedback.

Considering search result pages, we assume that a user does not examine all the documents in the result list and is more likely to examine higher ranked documents. Due

3. THEORETICAL BACKGROUND

to this position bias, we do not observe the relevance of all documents with an equal probability. More formally, the relevance of a document d on position k for query q is revealed with a probability p_k depending on the rank of the document in the result list. This is called the propensity of observation or simply click propensity. For counterfactual evaluation and learning employed in unbiased learning-to-rank we need to know this propensity of observation p_k . The click propensity can be used to re-weigh user interactions through Inverse Propensity Scoring (IPS) in the evaluation metric and loss function of the learning-to-rank model. To illustrate how the click propensities are employed in the unbiased learning-to-rank framework we give an example of an unbiased version of the Mean Reciprocal Rank (MRR) metric. The reciprocal rank is the multiplicative inverse of the first relevant result. Taking the average over all queries results in the MRR as formalized in 3.17 where k_i denotes the rank of the clicked document in the evaluated ranking. We can obtain an unbiased metric by defining a weighted MRR as formalized in 3.18 where w_k is the inverse propensity $1/p_k$ of the position on which the document was displayed during logging. We can observe from these equations that clicks with a high propensity are weighed less and clicks with low propensity are weighed higher. Intuitively, click on high ranks are more likely, thus weighed less, while clicks on lower rank are less likely and thus are weighed more. This counterfactual evaluation method can be extended to obtain an unbiased learning objective for learning-to-rank loss functions but we will not discuss details on this.

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{k_i} \quad (3.17)$$

$$MRR = \frac{1}{\sum_{i=1}^N w_k} \sum_{i=1}^N w_k \frac{1}{k_i} \quad (3.18)$$

In the unbiased learning-to-rank framework the Position-Based Propensity Model is introduced. The assumptions of the Position-Based Propensity Model [3] or Position Bias Model [13] are analogous to the Position-Based Model as discussed in Section 3.4.3. As formalized in 3.13, in this model, a click depends jointly on the probability of examination and the probability of relevance. In this model — under the assumption of noise free click log data — it is assumed that a clicked document is relevant. However, the reverse of this assumption does not always hold. A non-click or skip of a document does not necessarily indicate non-relevance as the user might not have examined the document. Thus, we do not observe the relevance for this document and the user interaction data is biased due

3. THEORETICAL BACKGROUND

to this unobserved feedback. In the Position-Based Propensity Model we assume that the propensity of observation p_k is equivalent to the probability of examination θ_k [3, 13].

$$p_k \Leftrightarrow \theta_k \tag{3.19}$$

We will adopt the naming conventions of Chuklin et al. [17] and Agarwal et al. [6] and refer to the models as the Position-Based Model and the Position-Based Propensity Model respectively.

As discussed in Section 3.4, there is an extensive range of more complex click models that also model position bias in different ways. However, it is an open question in how far these more complex click models can be adapted and employed for effective propensity estimation [13]. Prior works shows that the Position-Based Model is as effective as more complex click models in case of a single click per search session and high query frequency [36]. Li [19] consider position bias estimation for unbiased learning-to-rank in the context on hotel search and incorporate a similar assumption to the Cascade Model [17] that a user examines all the preceding documents $D_{<k}$ of the clicked document d on position k . Fang et al. [37], Agarwal et al. [38] propose to extend the unbiased learning-to-rank framework for context dependent propensity. The introduced Contextual Position-Based Model (CPBM) extends the Position-Based Propensity Model by including the context of a search query and the user. In this model the propensity can also depend on observable features of the query, document or user. Vardasbi et al. [39] introduce a Cascade IPS Model for unbiased learning-to-rank. The model is proven to outperform the PBM in case of cascade model user behavior.

In the unbiased learning-to-rank framework the learning objective is invariant to multiplicative scaling [2, 37, 38]. Thus, it is sufficient to estimate the propensities up to some positive multiplicative constant and p_k is estimated relative to position one, obtained through the relative ratio $\frac{p_k}{p_1}$. Most click models and the Position-Based Model specifically [17] incorporate a similar assumption that the document at position one k_1 is always examined.

3. THEORETICAL BACKGROUND

Table 3.1: Table of notation used in thesis.

Notation	Description
d	document identifier, product identifier
q	query identifier, search term
k	position, rank
c	click, add to basket event
b	binary order indication
a	binary re-buy indication
v	binary display event
u	user, customer
D	set of documents
Q	set of queries
S	set of search sessions
s	search session
S_k	set of search sessions with result on position k
$S_{q,d}$	set of search sessions for query q containing document d
(q, d, k, c)	click log item
f	ranking function
R	random variable for perceived relevance of document snippet
E	random variable for examination
\tilde{R}	random variable for true relevance
θ_k	probability of examination (position bias) of position k
$\gamma_{q,d}$	query-document relevance score
p_k	propensity of observation for position k

Chapter 4

Related Work

In this section the related work on position bias estimation for unbiased learning-to-rank is discussed. The considered methods focus on position bias estimation under the assumptions of the Position-Based Propensity Model. The key problem of position bias estimation is that we only observe the clicks of a user directly, yet the examination and relevance are never observed individually. Thus, the click data depends jointly on the query-document relevance and position bias which are both unobserved quantities. The high level goal of position bias estimation is to disentangle the effects of the unobserved quantities on the only observed quantity, the clicks. Two types of position bias estimation methods can be distinguished. Methods that rely on online intervention and methods that estimate position bias using historical click log data. The latter can be divided into methods that rely on explicit modeling of query-document relevance — which can be a difficult problem in many scenarios — and methods that do not. In this chapter we will answer the following sub-question:

RQ2: Which methods for position bias estimation exist in the literature and how do they compare?

4.1 Result Randomization Methods

Result randomization methods are online intervention methods that aim to estimate the position bias directly without explicitly modeling and learning the query-document relevance. The core concept of result randomization is to permute the ranked result list consisting of n documents for a given query and present the permuted list to the end user instead of the original result list. The permutation is employed through a randomized experiment where the assignment of a position for a document does not depend on the

4. RELATED WORK

relevance or any covariates. This procedure is applied to all searches for a (small) fraction of the end users and the observed results can be used to estimate position bias as the change in click through rates is proportional to the difference in examination. The method proposed in [2] was the first result randomization method for position bias estimation. The method uniformly at random permutes the complete result set of size n . Yielding an equal probability for each document of being displayed at position k , resulting in a uniform distribution of the query-document relevance over all positions $k \in n$. Thus, the position bias for position k can be obtained by taking the observed click through rate on the corresponding position in the results of the randomized experiment.

An extension to this method is the *Randomize TopN* method proposed by Wang et al. [13]. In this method only the top $N \in n$ documents for which position bias estimation is desired are permuted. Both uniformly randomizing the complete result list or only the top N are intrusive interventions as they degrade retrieval performance. Thus, the methods have a negative impact on the user experience. The following result randomization methods aim to reduce the impact on the retrieval performance by introducing less intrusive permutation interventions.

Joachims et al. [3] propose a result randomization method named *SwapK*. The authors prove that in the unbiased learning-to-rank framework it suffices to estimate the propensities up to some positive multiplicative constant. Thus, it is only required to estimate the position bias p_k relative to position 1 (or another anchor position), obtained through the relative ratio $\frac{p_k}{p_1}$. This is related to the general assumption in click models that the first position is always examined by the user. The method defines the following intervention, randomly swap the documents at rank 1 and rank k with a fixed probability p such as $p = .5$. The intervention must be performed for any position k for which the position bias effect is desired to be estimated. The interventions of this method are less intrusive compared to the *Randomize TopN* interventions. However, swapping the item in the first position to an arbitrary lower ranked position is in many cases, especially for large k , still an intrusive intervention. Considering the fact that in the case of a proper ranking system the document on the first position is one of the most relevant documents in the result list. The *Randomize Pair* method proposed in [13] defines the least intrusive interventions of the different result randomization approaches. Similar to the *SwapK* intervention, but only swapping adjacent pairs, the interventions randomly swaps documents on adjacent pairs at position k and $k - 1$. Based on these interventions we can obtain the relative ratio $\frac{p_k}{p_{k-1}}$ between position k and $k - 1$. Under the assumptions of the considered Position-Based Propensity Model discussed in Section 3.5, we can obtain the relative ratio between

4. RELATED WORK

any two position by multiplying the ratios of the adjacent positions 4.1. The method is different compared to the *Randomize TopN* as the expected relevance can differ between position pairs and thus the relevance is not constant over all positions. Nonetheless, given the assumptions of the position bias model the relative ratios between different positions are the same as with the *Randomize TopN* method. The *Randomized Pair* may need to run longer to get a sufficient amount of clicks for position bias estimation as the lower positions receive less clicks. There is no existing research that considers how much traffic is required for accurate position bias estimation using the different result randomization methods.

$$\frac{p_k}{p_1} = \frac{p_2}{p_1} \cdot \frac{p_3}{p_2} \dots \frac{p_k}{p_{k-1}}. \quad (4.1)$$

Another option to reduce the negative impact of the interventions of the *SwapK* and *Randomize Pair* is to estimate position bias for specifically selected fixed ranks and apply interpolation as suggested in [3]. A similar approach is applied in [20] but here the fixed rank propensities are not estimated using results randomization.

Result randomization is the most accurate position bias estimation method and is considered the gold standard when compared to other methods. However the major drawback of the different result randomization methods is that the interventions degrade retrieval performance and consequently have a negative impact on the user experience. Wang et al. [13] empirically compare the *Randomize TopN* and *Randomize Pair* methods with respect to the effectiveness of the position bias estimation as well as the negative impact on the ranking performance. A portion of the traffic of two search services is compared, an online email (Gmail) and file storage service (Google Drive). The position bias effect is estimated for the first five positions and both methods align well. The negative effect on the search experience is quantified by means of the relative difference in the Mean Reciprocal Rank metric between the randomized and production search traffic. The results show that both methods significantly decrease the user experience with respect to the MRR metric on both services. However, the less intrusive interventions of *Randomize Pair* result in a lower negative impact compared to the *Randomize TopN*.

4.2 Historical Click Log Methods

The following methods aim to estimate position bias from historical click log data. The methods exploit randomization — organic or explicit — present in the historical data.

4. RELATED WORK

These methods do not rely on online interventions and thus do not degrade retrieval performance and user experience. Furthermore, most search systems have click log data in abundance readily available. Two types of these methods can be distinguished, methods that rely on explicit modeling of the query-document relevance and methods that do not.

4.2.1 Standard Expectation-Maximization

The traditional approach to learn the position-based model parameters is through a generative approach. To estimate position bias, one needs to estimate the latent variables of the position-based model which are both the examination probability per position θ_k and the query-document relevance $\gamma_{q,d}$. Thus, this method relies on relevance modeling. The classic approach employs a Bayesian network for which the parameters can be estimated using the Expectation-Maximization algorithm [17, 18]. The EM algorithm can find the latent parameters that maximize the likelihood of the historical click log S by iteratively performing the Expectation and Maximization steps. In the Expectation step, the model parameter values are assumed to be fixed and given these variables from the previous time-step the distribution of the hidden variable E and R are estimated. In the Maximization step the new parameter values are derived using the quantities from the Expectation step. The drawback of this approach is that it requires queries to repeat many times for reliable relevance estimation and a document must appear on a sufficient number of different positions for a reliable estimation of position bias [13, 17, 30].

4.2.2 Regression Based Expectation-Maximization

Wang et al. [13] propose a regression based EM algorithm to estimate the latent parameters of the PBM. The regression based EM algorithm aims to overcome shortcomings of the standard EM method, specifically in the context of personal search. In personal search the q, d identifiers may not be available due to privacy limitations. Furthermore, the document collection is private and the click data is highly sparse. The method introduces a modification of the Maximization step. Instead of using the exact query-document identifiers q, d , the method involves a feature vector $\mathbf{x}_{q,d}$ that represent the query-document pairs. The feature vectors are used to estimate relevance using a function $\gamma_{q,d} = f(\mathbf{x}_{q,d})$. The objective of the Maximization step is to find a regression function $f(\mathbf{x})$ that, given the estimations from the Expectation step, maximizes the log-likelihood. Specifically, for each data point $(c, q, d, k) \in s \in S$, we have feature vector $\mathbf{x}_{q,d}$ and binary relevance $r = 0, 1$ —

4. RELATED WORK

effectively turning the problem into a classification problem — and the $P(R = 1|c, q, d, k)$ is estimated by the regression function $f(\mathbf{x}_{q,d})$. The objective of $f(\mathbf{x})$ is formalized in 4.2.

$$\sum_{\{\mathbf{x},r\}} r \cdot \log(f(\mathbf{x})) + (1 - r) \cdot \log(1 - f(\mathbf{x})) \quad (4.2)$$

This method does not require queries to repeat or for documents to appear in multiple different positions. It merely requires that similar feature vectors appear in multiple different positions. The authors suggest that the feature vector $\mathbf{x}_{q,d}$ can be the same as the ranking features used by the current production ranking system. Consequently, this method can be efficiently adopted in systems where ranking features and click log data are collected. Wang et al. [13] performed experiments on datasets of both a personal email and file storage service. The results show that the method provides less accurate position bias estimations compared to the result randomization method *Randomize Pair* as the position bias effect is overestimated. However, the estimations still provide a significant improvement of ranking performance for the considered unbiased learning-to-rank model.

4.2.3 Embedded in Discriminative Models

This method employs discriminative models and aims to estimate position bias by embedding the position as an input feature for the discriminative model that predicts click probabilities (similar to clicks models). Similar to the regression based EM discussed in Section 4.2.2 we have a feature vector \mathbf{x} , to which the encoded position is appended. Yielding a training instance $([k, \mathbf{x}_{q,d}], c)$ for each click log item $(c, q, d, k) \in s \in S$. A discriminative model can be trained to predict click probabilities using this data. Subsequently the position bias effect must be separated from the other ranking features. Wang et al. [13] propose to employ a Gradient Boosted Decision Tree (GBDT) and separate the position effect by setting the split depth to 1 which disallows feature interactions. In this method the position is treated like a regular feature. Therefore, the click probability can be attributed to both the position feature or any of the other ranking features. The difficulty with this method lies in the separation of the position bias effect and the ranking features (relevance). The generative approach discussed in Section 4.2.2, that employs a probabilistic graphical model, has a clear separation of the position bias and relevance component. This structure allows learning of each component independently. Conversely, with this method the separation of effects is a difficult problem. Wang et al. [13] show that the embedded method provides less accurate position bias estimation compared to the regression based EM.

4. RELATED WORK

4.2.4 Intervention Harvesting

The drawback of the aforementioned methods is that they rely on explicit relevance modeling. Defining an accurate relevance model is in itself a difficult problem, that is in many cases just as difficult as the learning-to-rank problem. Intervention Harvesting [6, 38] aims at exploiting natural interventions in the historical click log data for position bias estimation and avoids explicit relevance modeling. These natural interventions are present in the click log data of ranking systems that deploy multiple rankers at the same time in A/B tests or where the production ranker is updated frequently. The method is fundamentally interventional and thus analogous to methods that rely on explicit interventions. However, unlike the results randomization methods, Intervention Harvesting solely relies on natural interventions and requires no additional online interventions. Furthermore, the method does not rely on relevance modeling and there is no prerequisite for queries to occur multiple times.

The method relies on harvesting interventions from historical clicks log data under mild assumptions. Considering a historical click log S with rankings from m historical rankers $F = \{f_1, \dots, f_m\}$. A crucial condition for this method, as formalized in 4.3, is that the query and user distribution do not depend on the ranker (i.e. rankers should not handle different types of queries or users) (i.e. the choice of f_i does not depend on q). This generally holds for rankers that are compared in A/B tests. However, rankers that are deployed in sequence of production might introduce a temporal covariate shift in query and user distribution. This shift should always be guaranteed to be small for this method to be valid.

$$\forall f_i : P(Q|f_i) = P(Q) \implies \forall q \in Q : P(f_1|q) = P(f_1) \quad (4.3)$$

Interventional sets $S_{k,k'}$ of query-document pairs are constructed by harvesting sets from the click log where two ranking functions f and f' rank a document d on positions k and k' where $k \neq k'$ for the same query.

Formally, for a fixed number of top position M for which position bias estimation is considered, for each two ranks where $k \neq k' \in M$ the interventional sets are defined as 4.4

$$S_{k,k'} := \{(q, d) : q \in Q, d \in D, \exists f, f' \text{rank}(d|f(q)) = k \wedge \text{rank}(d|f'(q)) = k'\} \quad (4.4)$$

The sets contains two different treatments as the rank of d was randomly assigned via the choice of rankers which is a random intervention under the assumption of 4.3. However, the

4. RELATED WORK

position assignment is generally not uniform and thus weights that reflect the probability of a document being ranked at position k are applied to account for this non-uniformity.

Unlike the result randomization methods, where the interventions are explicitly controlled, the swap interventions in Intervention Harvesting are not. Consequently we might not observe direct swaps between all the ranks considered for position bias estimation. If we take the example of *SwapK* it might be the case that no swaps between position 1 and some arbitrary position k are observed in the click log. To solve this, Agarwal et al. [6] propose a method to aggregate many local swaps into an overall estimate of position bias for any considered k named the *Global AllPairs Estimator*. Agarwal et al. [6] consider three test scenarios where all results show estimations consistent with the gold standard.

4.2.5 Dual-Learning Algorithm

Ai et al. [40] propose a dual-learning algorithm that jointly learns an unbiased ranker and the position bias effect. The method defines the Inverse Relevance Weighted (IRW) loss function, similar to the inverse propensity weighted loss. The method employs a combination of both the IPW and IRW loss functions to simultaneously learn an unbiased learning-to-rank model and the position bias effect. The advantage of the method is that we have a fully automatic learning process. However, this also introduces a tight coupling between the position bias estimation and learning-to-rank.

4.2.6 Simplified Likelihood

Aslanyan and Porwal [20] propose a method to directly estimate position bias without modeling relevance in the context of e-commerce. The method involves harvesting query-document pairs that appeared at multiple different positions and a likelihood function that only depends on position bias and not the query-document relevance. The likelihood function is derived under assumptions on the considered click log data and can be used to directly estimate position bias. The method assumes that query-document pairs appear a few times on different positions and receive at least one click and one non-click. Furthermore, it is assumed that the click probabilities for a query-document pair ranked on an arbitrary position k are low. Aslanyan and Porwal [20] conduct an experimental evaluation of the proposed methods through two experiments. First, the estimated position bias is compared to the true position bias using a simulated dataset. Second, an unbiased learning-to-rank model is trained for eBay search data using the estimated position bias and the model performance is compared to a biased baseline as well as the EM method

4. RELATED WORK

discussed in Section 4.2.2. Unlike other studies, Aslanyan and Porwal [20] do not conduct an evaluation of the proposed method through a comparison of the bias estimated using the proposed methods and using a result randomization experiment.

Chapter 5

Methodology

In this chapter we discuss the methodology of this thesis. First, we address the question of which method for position bias estimation can be employed in the context of Picnic. Next, we discuss the model we propose to implement and how we propose to adapt and extend the models. Furthermore, we describe the experimental setup of the thesis consisting of the data pre-processing, a detailed description of implemented (baseline) models and the evaluation. Finally, we describe the proposed extensions of the models considering noise filtering, display events and personalized ranking.

5.1 Position Bias Estimation Method

The related work discussed in Chapter 4 proposes different methods for position bias estimation that are applied in various scenarios. Based on the findings on the related work and the characteristics of the Picnic in-app search engine and the corresponding data, we will answer the following sub-question:

***RQ3:** Which method for position bias estimation can we employ and adapt in the case of Picnic?*

Although proven as the most reliable and consistent method for position bias estimation, result randomization methods [2, 13] have several disadvantages over the historical click log based methods. First, the methods degrade retrieval performance to varying degrees. Consequently, resulting in a negative impact on user experience and potential loss in revenue. In the case of Picnic and e-commerce in general both drawbacks are not desired and can be costly. Second, the result randomization methods require an extensive testing setup

5. METHODOLOGY

that involves the production ranking system. Introducing modifications to production systems solely with the purpose of experimentation is both costly and not desired. Third, as the ranking system or user population evolves, re-estimation of position bias requires a new randomization intervention. Furthermore, the position bias estimation is limited to the context of the experiment and to estimate position bias for varying context, additional interventions are required. Methods that rely on historical click log data can be exploited to learn models that account for changes in position bias due to context without additional interventions.

We aim at an independent position bias estimation method of which the results can be employed in any (future) unbiased learning-to-rank model or other use case. Thus, the Dual-Learning algorithm proposed by Ai et al. [40] does not fit this scenario as the learning of position bias and an unbiased ranking model are tightly coupled.

The intervention harvesting method [6] is proven to yield a similar accuracy to the result randomization methods. However, the intervention harvesting method assumes that the natural interventions in the data are the result of using multiple rankers in the historical ranking system. Furthermore, the harvesting of interventions has strict assumptions that either the pairs should be harvested from A/B testing results or an equal distribution of queries and users over the rankers is guaranteed. Furthermore, this method assumes that the relevance of a document does not change much over time which is an unrealistic assumption for many e-commerce scenarios. The similar simplified likelihood method proposed by Aslanyan and Porwal [20] has no restrictions on the query and user distribution but has strict assumptions on the product pairs and the click-through rates on the ranks. In the case of Picnic we cannot ensure that all the assumptions and requirements of the aforementioned methods can be met.

In the context of Picnic and similar e-commerce stores we can — with certain adjustments — estimate position bias through the standard EM algorithm by leveraging some inherent properties of the ranking system and the resulting historical click data. We have a relatively small non-private collection of documents and we can work directly with the q, d identifiers unlike [13]. Furthermore, while the requirement of repeating might be unrealistic in many cases, in the case of an online grocery store this is not the case. We have many repeating queries that receive a lot of clicks, even by the same customers. As described by Agarwal et al. [6], Aslanyan and Porwal [20] defining an accurate relevance model — which we need to learn with this approach — can be a difficult problem due to the large number of parameters. However, we focus on position bias and are not directly interested in learning the query-document relevance $\gamma_{q,d}$. Though, necessary to estimate the

5. METHODOLOGY

relevance parameters $\gamma_{q,d}$ they be considered nuisance parameters. This allows us to focus on head queries and thus reduce the number of parameters. Furthermore, we have a large quantity of user interaction data for these head queries which simplifies the learning of the relevance model. Moreover, we can leverage the natural or organic randomization present in the click log data. (q, d) pairs appear at many different positions due to the dynamic nature of e-commerce ranking. We propose to estimate position bias estimation under the Position-Based Propensity Model learned through the standard Expectation-Maximization algorithm solely utilizing queries for which we have sufficient user interactions.

The search sessions in the Pinic dataset mostly have a single click or no click, as observed in Table 2.3. Furthermore, we focus on head queries that have a high query frequency. As discussed in Section 3.4, the PBM has been shown to as effective as more complex click models in this scenario. Thus, we do not propose to investigate the use of an alternative to the Position-Based Propensity Model.

5.2 Models

In this section we formulate the models employed in this thesis. We first briefly discuss the baseline models. Next, we extensively discuss the Position-Based Propensity Model and the Expectation-Maximization algorithm used for parameter estimation.

5.2.1 Baseline Models

We implement two baseline models, the Document-Based CTR Model and Rank-Based CTR Model. The model are used as a performance baseline for the more complex Position-Based Propensity Model. Furthermore, as discussed in detail in Section 5.2.2, the parameters values of the models are used to derive initial values for the Expectation-Maximization algorithm. Both the DCTR and RCTR model comprise exclusively observed variables. Thus, we can estimate the parameter values through maximum likelihood estimation [17]. The parameters are estimated using equations formalized in 5.2 and 5.1 for the DCTR and RCTR model respectively. We introduce a modification to the MLE of the RCTR model to accommodate for the variable number of result documents (not all search sessions have the same number of results). In the algorithm as described by Chuklin et al. [17] we take the total number of sessions $|S|$. However, we only consider the number of sessions in S that include a document on position k denoted S_k .

$$\rho_k = \frac{\sum_{s \in S_k} \sum_{(c,q,d,k) \in s} c}{|S_k|} \quad (5.1)$$

5. METHODOLOGY

$$\rho_{q,d} = \frac{\sum_{s \in S_{q,d}} \sum_{(c,q,d,k) \in s} c}{|S_{q,d}|} \quad (5.2)$$

5.2.2 Position-Based Propensity Model

We implement the Position-Based Propensity Model to estimate position bias and infer the (latent) parameters employing the standard Expectation-Maximization algorithm described by both [2, 17]. Expectation-Maximization can be considered an approach of the maximum likelihood estimation algorithm for models that comprise latent variables. The Position-Based Propensity Model is such a model in which we have latent parameters E_k and $R_{q,d}$. The objective of the EM algorithm is to find the parameters that maximize the log-likelihood \mathcal{LL} of the model given observed click log S . The intuition behind the model is to take a bootstrap approach and initialize the model with a set of parameter values, either arbitrary values or following some prior knowledge on the parameter values. Next, we assume the parameter values to be known and complete the data by estimating the latent or hidden variables for every data point. Then, we treat the completed data set as observed and learn a new set of parameters through maximum likelihood estimation. The process of completing the data (Expectation) and estimating a new set of parameters (Maximization) is repeated until convergence.

More formally and specifically for the Position-Based Propensity Model. Consider independent random Bernoulli variables for relevance $R_{q,d}$ and examination E_k , with corresponding latent parameters θ_k and $\gamma_{q,d}$ respectively, as formalized in 3.13. We have a click log S where $S_{q,d}$ and S_k denote the set of sessions that contain a query-document (q, d) pair and that consider a position k respectively. The objective of the EM algorithm is to find the parameters θ_k and $\gamma_{q,d}$ that maximize the log-likelihood \mathcal{LL} of the model given the observed click log S . The model starts with initialization of the parameters with some starting point value. We discuss details on the initial values later in this section as well as in Section 5.5.2. Subsequently, the EM algorithm iteratively alternates over the Expectation and Maximization steps. In the Expectation step the distribution of latent variable E and R are estimated for each click log item $(q, d, k, c) \in s \in S$ using formulas 5.3, 5.4 and 5.5. The latent variables are estimated using the observed variables of the corresponding click log item and the parameters $\gamma_{q,d}$ and θ_k , from the previous iteration.

$$P(E = 1, R = 1 | C = 1, q, d, k) = 1 \quad (5.3)$$

5. METHODOLOGY

$$P(E = 1, R = 0 | C = 0, q, d, k) = \frac{\theta_k(1 - \gamma_{q,d})}{1 - \theta_k\gamma_{q,d}} \quad (5.4)$$

$$P(E = 0, R = 1 | C = 0, q, d, k) = \frac{(1 - \theta_k)\gamma_{q,d}}{1 - \theta_k\gamma_{q,d}} \quad (5.5)$$

From the completed data where the distribution of E and R are estimated for every click log item, the expected sufficient statistic for the parameters $\{\theta_k\}$ and $\{\gamma_{q,d}\}$ are calculated using 5.7 and 5.6 respectively.

$$ESS_{\gamma_{q,d}} = \mathbb{E} \left[\sum_{s \in S_{q,d}} \sum_{(c,q,d,k) \in s} (c_d + (1 - c_d) \cdot P(E = 0, R = 1 | C = 0)) \right] \quad (5.6)$$

$$ESS_{\theta_k} = \mathbb{E} \left[\sum_{s \in S_k} \sum_{(c,q,d,k) \in s} (c_k + (1 - c_k) \cdot P(E = 1, R = 0 | C = 0)) \right] \quad (5.7)$$

In the subsequent Maximization step we assume the expected sufficient statistics as observed and derive new values for the parameters $\{\theta_k^{t+1}\}$ and $\{\gamma_{q,d}^{t+1}\}$ by performing maximum likelihood with respect to the expected sufficient statistics. We derive the new optimal values that maximize the likelihood for parameters $\{\theta_k\}$ and $\{\gamma_{q,d}\}$ using 5.9 and 5.8 respectively.

$$\gamma_{q,d} = \frac{1}{|S_{q,d}|} \cdot ESS_{\gamma_{q,d}} \quad (5.8)$$

$$\theta_k = \frac{1}{|S_k|} \cdot ESS_{\theta_k} \quad (5.9)$$

The algorithm is formalized in 1. We introduce a modification of the standard EM algorithm [13, 17] to tailor it to the scenario of Picnic. We have a variable number of products in the search results page thus instead of S , we have S_k in 5.9.

In order to initiate the Expectation-Maximization algorithm initial values for the parameters $\{\theta_k\}$ and $\{\gamma_{q,d}\}$ are required. As suggested by [17] we utilize the output of a simpler model — the baseline models — to initialize the parameters values. First, we take the parameter values $\rho_{q,d}$ of the DCTR model as initial values for $\gamma_{q,d}$. The parameters values ρ_k of the RCTR model are normalized by the CTR of the first position ρ_1 — as we aim to estimate relative propensity — and used as initial values for θ_k . We will refer to this as the *normalized empirical CTR*.

5. METHODOLOGY

The iterative process of the EM algorithm is theoretically guaranteed to improve the likelihood upon previously derived parameters on every iteration. The algorithm is considered to converge if no further improvements can be made to the likelihood. It is common practice to conclude the algorithm before complete convergence to reduce computational time [41], considering the fact that the convergence rate of the likelihood is fast in the first few iteration but slows down from a certain iteration to complete convergences [17]. The likelihood improvements will be minimal from this iteration on. Thus, we propose to concluded the algorithm at a fixed threshold number of iterations instead of running until complete convergence. We discuss the process on finding this threshold value in Section 5.5.1.

5.3 Data Pre-Processing

We split the data into bucket of one day in which all click log items for the given day are captured. We do this to ensure the chronicle order of the events as well as to ensure efficient loading and preprocessing of the data, by avoiding to load every individual event in the downstream analysis pipeline.

Search sessions without clicks do not contribute to the learning objective of Position-Based Propensity Model. Hence, we consider only search sessions with at least one click. More formally, we consider sessions $S_{c=1} = \{s \in S \mid (\exists(q, d, k, c) \in s)[c = 1]\}$. Furthermore, a search session can contain multiple add to basket event for a single product. The customer can add multiple units of a single product to the basket. We consider only a single add to basket event per product per search session as a higher quantity of purchase does not indicate higher relevance of a product for the given query.

We filter out any sessions that have missing values for any of the required click log item values. Furthermore, we filter out any queries for which the result set size N , over all queries in the training set $N \leq 1$. The queries with only a single item in the result set do not contribute to the learning objective of the model and induce unnecessary compute.

The primary goal of the Position-Based Propensity Model is position bias estimation. The learning of a relevance model for the complete set of queries as well as click predictions, are secondary requisites. Queries that do not repeat a sufficient amount of times do not contribute to the position bias estimation. For non-repeating queries the model cannot observe changes in clicks due to variations in the position of products. Hence, we filter out any non-repeating queries.

5. METHODOLOGY

Algorithm 1: Position Based-Propensity Model Expectation-Maximization

```

input :  $S, \{\rho_k\}, \{\rho_{q,d}\}$ , iterations
output:  $\{\theta_k\}, \{\gamma_{q,d}\}$ 
1 Initialize Parameters
2  $\{\theta_k\} \leftarrow \{\rho_k\}$ 
3  $\{\gamma_{q,d}\} \leftarrow \{\rho_{q,d}\}$ 
4  $t \leftarrow 0$ 
5 while  $t < \text{iterations}$  do
6   Expectation Step
7   Estimate distribution of latent parameters E and R for every click log item
8   Calculate ESS for each parameter based on estimated distributions.
9    $\text{ESS}_\theta[\text{sum}_k] \leftarrow 0$ 
10   $\text{ESS}_\gamma[\text{sum}_{q,d}] \leftarrow 0$ 
11  foreach  $s \in S$  do
12    foreach  $(q, d, k, c) \in s$  do
13      if  $c = 1$  then
14         $E = R = 1$  as per 5.3
15      else
16         $E = P(E=1, R=0 | C=0, q, d, k)$  estimate using 5.4
17         $R = P(E=0, R=1 | C=0, q, d, k)$  estimate using 5.5
18      end
19       $\text{ESS}_\theta[\text{sum}_k] += E$ 
20       $\text{ESS}_\gamma[\text{sum}_{q,d}] += R$ 
21    end
22  end
23  Maximization Step
24  foreach  $\theta_k \in \{\theta_k\}$  do
25     $\theta_k = \frac{1}{|S_k|} \cdot \text{ESS}_\theta[\text{sum}_k]$ 
26  end
27  foreach  $\gamma_{q,d} \in \{\gamma_{q,d}\}$  do
28     $\gamma_{q,d} = \frac{1}{|S_{q,d}|} \cdot \text{ESS}_\gamma[\text{sum}_{q,d}]$ 
29  end
30   $t = t + 1$ 
31 end

```

5. METHODOLOGY

5.4 Evaluation

Both the position bias and query-document relevance are latent parameters of the Position-Based Propensity Model. Hence, the estimated position bias cannot be evaluated directly as the true position bias — which we aim to estimate — is unknown. However, as we estimate position bias under the Position-Based Propensity Model we can evaluate the fit of the model on the observed data by means of the likelihood function. The log-likelihood functions as formulated in Section 3.4 can be used to evaluate the model fit on a hold-out test set. The log-likelihood scores can be used to compare different model with respect to the fit on the test set. This evaluation metric is an indication on how accurate the inference of the latent parameters is. Thus, the log-likelihood score is indicative of the accuracy of the position bias estimation. Though, the RCTR and DCTR models cannot be used to estimate position bias, we can use the log-likelihood scores of the models as performance baselines. Traditional clicks models can additionally be evaluated on perplexity [29, 32]. However, this is not a widely used metric — especially not in the recent unbiased learning-to-rank framework — and difficult to interpret [17]. Thus, we do not consider the perplexity metric for evaluation in this thesis. The log-likelihood score grows relative to the sample size. To compare the log-likelihood over test sets of varying sample sizes we take the average log-likelihood. The average log-likelihood can be computed with the formulas given in 3.10, 3.8, 3.16 for the RCTR, DCTR and PBM respectively [13, 17, 42]. The optimal value of the log-likelihood is 0, indicating a perfect click prediction and a larger value indicates a better performance with respect to model fit on the test data.

The click log data is intrinsically time ordered. To evaluate the model performance we take a time window $t - 1$ to fit the model and time window t to evaluate the model. This ensures that we test the model on out-of-sample observation which are in this case unseen future search sessions. Furthermore, this ensures that seasonality effects and emerging changes in customer behaviour patterns can be picked up by the model, which is important for the relevance modeling.

The model requires both sufficient randomization in the query-document positions (to learn the position bias) as well as sufficient query frequencies (to learn an accurate relevance model). The most trivial option would be to train the model on the complete dataset, or a substantially part of it, and hold-out a part of the data to test on. However, we need to consider a few aspects of the data and models that constrain us to use the complete dataset. First, we need to find a balance between the sample size and the long-term relevance variations. The Position-Based Propensity Model requires enough natural

5. METHODOLOGY

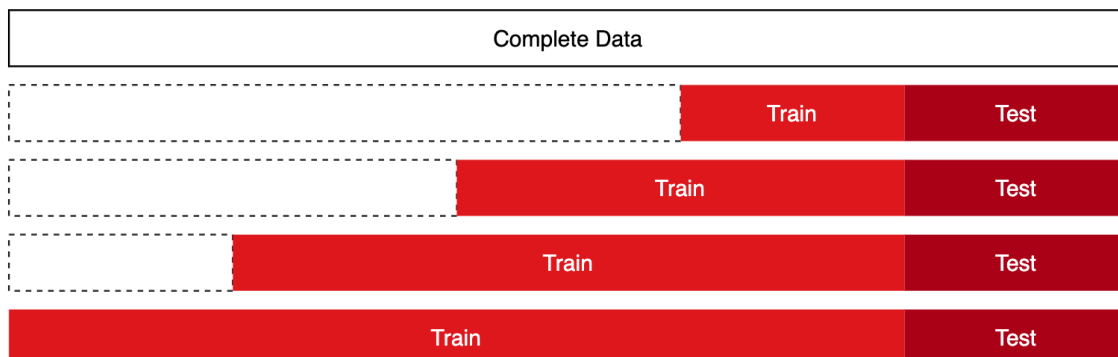


Figure 5.1: Window size experiment *expanding window* strategy.

randomization in the positions of document. However, in e-commerce, variations in the relevance of documents can occur over time due to variations in item popularity, seasonality effects, price fluctuations and other covariates [19, 20]. This will introduce randomization in positions due to variations in relevance. Secondary, this would yield a sample size of hundreds of millions of sessions which would drastically increase the computational complexity of the system. We propose the following setup to tackle this problem. First, we find a sample size (in buckets of days) on which the performance of the model converges. To find this sample size and adhere to the natural order we implemented the *expanding window* strategy [43], as depicted in Figure 5.1. The size of the dataset allows us to apply this to a separate sample of the complete dataset which can be seen as a validation set to avoid data leakage. The sample size is designated as the windows size in the subsequent experiments. Second, to evaluate the model generalization performance we implement a modified version of k-fold cross-validation [44, 45, 46] that adheres to the natural time order of the data. Furthermore, to ensure independence of the training samples we implement a *sliding window* strategy as depicted in Figure 5.2 similar to [17, 43]. In the absence of model hyper-parameters we do not implement a validation set in the *sliding window* strategy. We take a period starting in January 2019 ending in May 2020 resulting in 16 folds in total.

5.4.1 Hypothesis Testing

We employ statistical hypothesis testing to find if the differences in performance between the models are statistically significant. Like other research on position bias estimation and unbiased learning-to-rank [2, 4, 10, 47] we apply a two-tailed paired student t-test and take a significance level of $\alpha = 0.01$. Thus, we consider the difference in model

5. METHODOLOGY

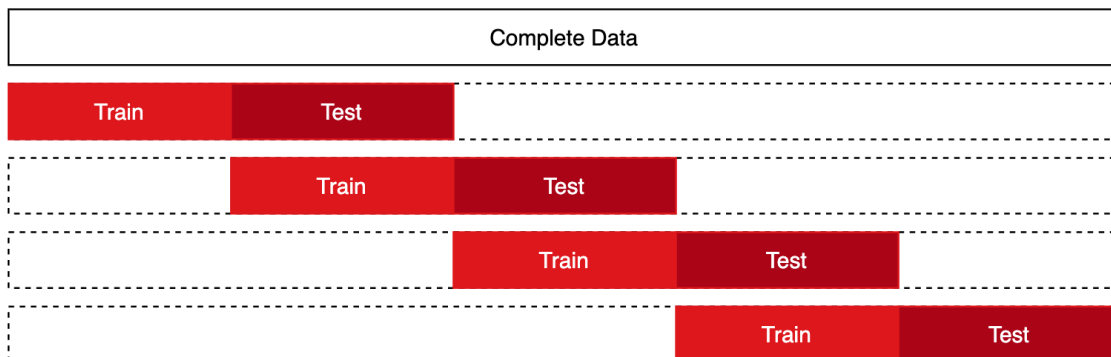


Figure 5.2: Time ordered k-fold *sliding window* strategy.

performance statistically significant with $p < 0.01$. We apply a paired t-test to compare the log-likelihood scores of the model on the same fold over all considered data windows. We can apply the t-test as, unlike standard k-fold, the training samples are independent over the data windows. The p-value is calculated by means of the t-distribution with the degrees of freedom set as the number of folds minus one [48]. Furthermore, we report the arithmetic mean of the model performances over the windows and the 95% confidence intervals based on the t-distribution, similar to Chuklin et al. [17]

5.5 Experiments

In this section we propose various experiments to conduct in this research. We first propose experiments on general improvements of the Position-Based Propensity Model. Next, we propose experiments with the aim of improving and extending the model utilizing Picnic specific characteristics and data to answer the following sub-question:

RQ4: How can we improve the position bias estimation method utilizing specific characteristics and data of the Picnic in-app search engine

5.5.1 Expectation-Maximization Iteration Threshold

As discussed in Section 5.2.2 we propose to conclude the iterative EM algorithm at a fixed threshold iteration. The literature on click models provides no specific recommendations on a strategy to find the number of iterations for the algorithm. Chuklin et al. [17] take a static value of 50, whereas Wang et al. [13] conclude the algorithm at 10 iterations. Considering the fact that the literature provides no recommendations and the fact that the Picnic dataset has different characteristics compared to the related literature. We aim

5. METHODOLOGY

to find the optimal value for the number of iterations by employing a convergence plot and test with a range of threshold values. The convergence plot depicts the log-likelihood improvement on the training set at each iteration and is used to find a threshold value that balances the computation time of the algorithm as well as a near complete convergence score.

5.5.2 Expectation-Maximization Initial Values

The Expectation-Maximization algorithm is only guaranteed to converge to a point of zero gradient, which is not necessarily a local or global optimum [17, 18, 25, 49]. Consequently, the model can get stuck at a saddle point. The initial values affect the convergence point of the algorithm. We propose to initialize the model with the parameter values obtained from the RCTR and DCTR models. However, to ensure that the model is converging to a global optimum we experiment with the initial values of the model. Derived from related work [17, 20], we experiment with the following initial parameters values $\{\theta_k\} = .2$, $\{\theta_k\} = .5$, $\{\gamma_{q,d}\} = .2$, $\{\gamma_{q,d}\} = .5$, 5.10 and 5.11.

$$\theta_k = \min\left(\frac{1}{\log k}, 1\right) \quad (5.10)$$

$$\theta_k = \frac{1}{(1 + \log k)} \quad (5.11)$$

5.5.3 Expectation-Maximization Data Smoothing

We also experiment with data smoothing as suggested by [17, 28, 30]. We assume a prior distribution $Beta(1, 1)$ for both parameters $\{\theta_k\}$ and $\{\gamma_{q,d}\}$. More specifically, two impression observations are included for each parameter, of which one receives a click and one a non-click. Consequently, the numerator in 5.8 and 5.9 starts at two (click and non-click) and the denominator starts at one (click). This data smoothing is involved with sparsity of the click log data and in the case of Picnic this could potentially smooth the estimations for both the lower positions that receive limited clicks and query-document pairs that receive limited clicks for a given query.

5.5.4 Noise Aware Position-Based Propensity Model

The examination hypothesis of the Position-Based Propensity Model in the unbiased learning-to-rank framework has a strict noise free assumption $C = 1 \implies R = 1, E = 1$. However, this is an unrealistic assumption as clicks can happen for various unexpected

5. METHODOLOGY

reasons [2, 6, 37]. Initial research on the unbiased learning-to-rank framework adopts the assumption of uniform click noise, under which the model is proven to be robust to noise and able to learn an unbiased ranker despite click noise [2, 3]. However, as shown by Joachims et al. [9], click noise cannot be assumed to be uniform due to trust bias. Recent research by Agarwal et al. [10] proposes to explicitly model click noise to overcome the problem of non-uniform click noise. Agarwal et al. [10] extend the Position-Based Propensity Model by introducing additional parameters that model the click noise. Furthermore, as discussed in Section 3.4, a click only reveals the perceived relevance R on the document snippet. As proven by [30, 34] there can be a substantial discrepancy between the perceived relevance R and true relevance \tilde{R} . Chapelle and Zhang [30] explicitly model both the perceived relevance and the true relevance in the Dynamic Bayesian Network Model.

In this thesis we propose a novel two fold solution for this unrealistic noise free assumption. First, we consider add to basket events only, which indicate the intention of the user to buy the product. This is a stronger relevance signal compared to clicks in the scenarios considered in the related work. Consequently, this reduces the difference between R and \tilde{R} . Furthermore, we utilize order data to filter out unintended or reverted add to basket events. More formally, we have extended click log S_b where each click log item $(q, d, k, c) \in s \in S_b$ is extended to $([q, d, k, c], b)$. $b = \{1, 0\}$ denotes if the respective product was eventually ordered. We extend the Position-Based Propensity Model by introducing the assumption that $b = 0 \implies C = 0$ as formalized in Algorithm 3. Thus, we do not explicitly model click noise but rather avoid to consider these confounding events by assuming a non-click on click log items where $b = 0$. Unlike web search and similar scenarios, in e-commerce this is possible as we know if the customer eventually ordered the clicked product. Furthermore, customers of Picnic mostly interact with the search results by directly adding them to the basket from the result list.

5.5.5 Display Events Position-Based Propensity Model

In the aforementioned models we solely utilize add to basket events. However, more extensive user interactions are captured on the search result page, as discussed in Section 2.3. We can utilize these events to extend the Position-Based Propensity Model and potentially improve parameter estimations, similar to [1, 32, 50]. We propose an extension of the Position-Based Propensity Model that can leverage display events of individual product tiles. This display event indicates that a product tile was presented in the visible part of the screen. The display events can potentially improve the model as they convey more information on the behavior of the user on the search result page. We propose to use the

5. METHODOLOGY

display event as an indication of examination. However, we cannot assume that a display event implies examination of the corresponding document. Conversely, we can assume that a non-display implies that the user did not examine the document as the user can never examine a document that was not in the visible part of the screen. More formally, we have an extended click log S_v where each item in session $s \in S_v$ is extended to $([q, d, k, c], v)$ where $v = \{1, 0\}$ denotes a display or non-display of the respective instance. We assume that $v = 1 \not\Rightarrow E = 1$ as a display event does not guarantee examination. Conversely, we assume that $v = 1 \Rightarrow E = 0$ as a user can never examine a document that was not in the visible part of the screen. We incorporate the assumptions into the Position-Based Propensity Model by adjusting the estimation of the distribution of latent variables E and R . We assume that $v = 1 \Rightarrow E = 0$ thus we replace the parameter value θ_k in 5.4 and 5.5 with 0 yielding equations 5.12 and 5.13 respectively. The examination step is adjusted employing these equations. In the case of $v = 0$ we replace 5.4 with 5.12 as the examination probability is assumed to be zero and we replace 5.5 with 5.13 as the relevance probability does not depend on the examination for $v = 0$. In the case of $v = 1$ we consider the standard update rules as formalized in 5.4 and 5.5. The lines 16 and 17 in the EM algorithm as formalized in 2 are adapted according to the aforementioned extension.

$$P(E = 1, R = 0 \mid C = 0, q, d, k) = \frac{0(1 - \gamma_{q,d})}{1 - 0\gamma_{q,d}} = 0 \quad (5.12)$$

$$P(E = 0, R = 1 \mid C = 0, q, d, k) = \frac{(1 - 0)\gamma_{q,d}}{1 - 0\gamma_{q,d}} = \gamma_{q,d} \quad (5.13)$$

5.5.6 Personalized Ranking

[REDACTED]

5. METHODOLOGY

[REDACTED]

[REDACTED]

[REDACTED]

Chapter 6

Implementation

In this chapter we describe the implementation details of the system. First, we motivate the choice of technology. Next, we describe the setup of the production ready system implemented for Picnic. Last, we describe how we ensure the scalability of the system. In this chapter we consider the following sub-question:

RQ5: How can we develop a production ready and scalable system for position bias estimation?

6.1 Position-Based Propensity Model

In the absence of an open source implementation of the Position-Based Propensity Model we developed an implementation of the model in Python. We employ Python as the programming language for the model as Python is a widely used programming language in the field of machine learning. Furthermore, Python is the language used by Picnic for all data science and machine learning projects. Notable is that the implemented model has no external dependencies and relies only on pure Python.

6.2 Production System

The primary objective of the system is position bias estimation. However, we employ the Position-Based Propensity Model that also learns a relevance model. Though, only for head queries, the query-document relevance scores have a substantial business value for Picnic as these scores can potentially directly improve the current ranking system. We discuss some details on the use case in Section 6.3. Thus, we aim to implement the system with the joint objective of position bias estimation and pointwise query-document relevance learning for

6. IMPLEMENTATION

head queries. The pipeline is implemented in Python with the exception of the *Load Data* and *Data Preprocessing* steps which are implemented by means of SQL queries that run in *Snowflake*, the Data Warehouse (DWH) system of Picnic. The system is documented using Python docstrings and an extensive readme. An overview of the system is depicted in Figure 6.1 and the different components are listed and briefly discussed below.

- *Load Configuration*: The pipeline has a set of configurable parameters such as the train/test split ranges and the threshold number of iterations for the EM algorithm. The configuration is loaded from a YAML file and set on initialization of the pipeline execution.
- *Loading & Preprocessing Data*: For computational efficiency, the data loading and preprocessing steps are combined and executed in the data warehouse. This significantly reduces the amount of data that is loaded into the downstream pipeline.
- *Train DCTR & RCTR*: The Document-Based CTR and Rank-Based CTR models are trained utilizing custom Python implementations. The models can be trained in parallel. However, due to the limited runtime of the models we do not implement parallel training of the models to avoid overhead and complexity of parallel execution.
- *Output Parameter Preprocessing*: The output parameters of the DCTR and RCTR model are processed to be used as initial parameters for the Position-Based Propensity Model.
- *Train Position-Based Propensity Model*: The PBM model is trained on the configured train data set, given the initial values and the configured number of iterations.
- *Model Evaluation*: The models are evaluated using the log-likelihood scores on the configured test data set.
- *Anomaly Detection*: The output parameters of the PBM and DCTR are analyzed to detect unexpected or erroneous values. This is done in order to avoid that the values will be loaded and utilized in customer facing production systems of Picnic.
- *Format Output Data*: The parameter values of the PBM and DCTR are formatted such that they can be stored in columnar tables.
- *Store Output Data*: The formatted parameters as well as meta-data on the system run are stored in the DWH of Picnic. The table definition is depicted in Table 6.2.

6. IMPLEMENTATION

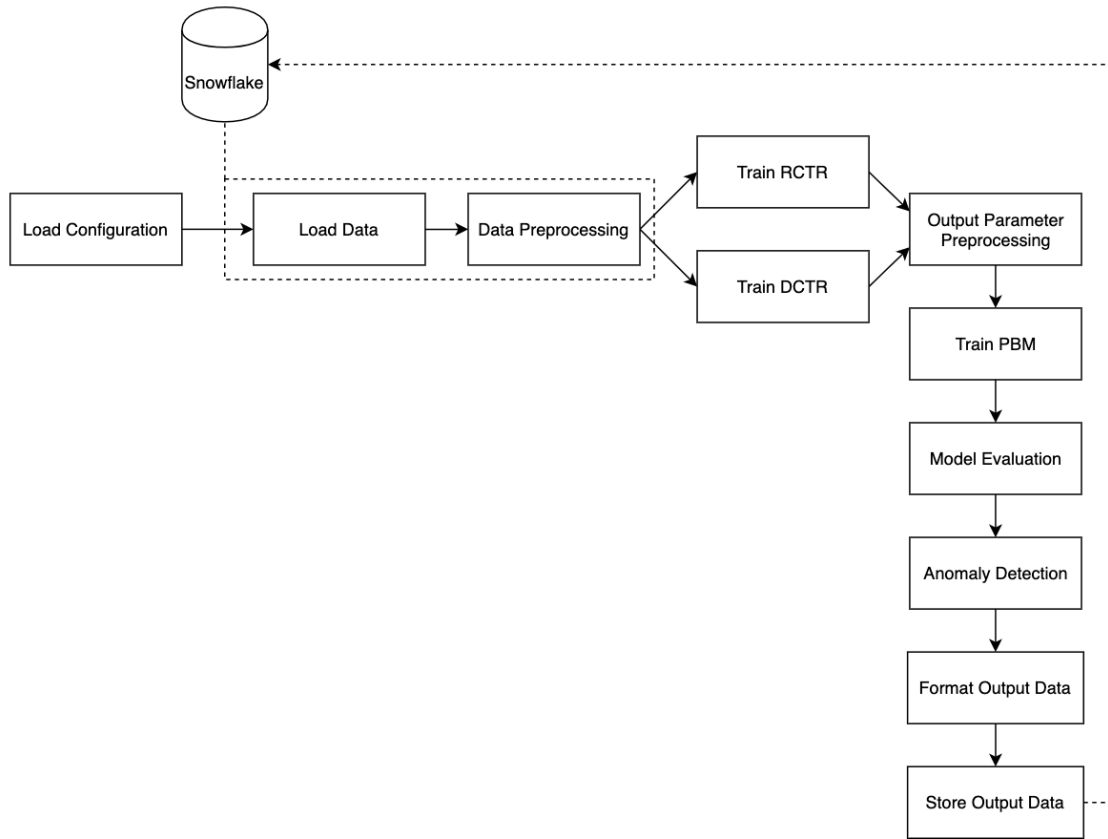


Figure 6.1: Overview of the production system.

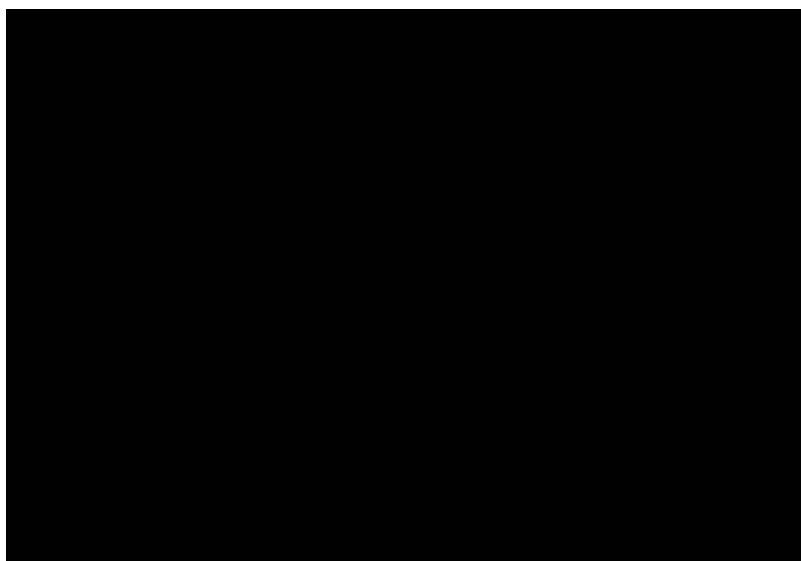


Figure 6.2: Overview of output tables in the Picnic data warehouse.

6. IMPLEMENTATION

6.2.1 Scalability

The data volume processed by the system will be in the tens of millions of click log items. We propose a two-fold solution to ensure scalability of the system. First, the data preprocessing component of the pipeline is executed in the data warehouse system of Picnic. The DWH system is optimized for loading of data from the various tables and automatically scales with the computational load of the preprocessing queries. Furthermore, preprocessing in the data warehouse will reduce the data volume that is loaded into the downstream pipeline components. Thus, reducing communication and network overhead. Second, though the preprocessing is done in the DWH, the downstream pipeline will still have to process tens of millions of click log items. Specifically, the training of the model is a potential bottleneck of system performance and scalability. To ensure scalability of the model training we do not employ a distributed computing system (e.g. Spark), but rather optimize the algorithm in such a way that it can be handled on small-scale systems and does not directly require data parallelism or a distributed system. We optimize the algorithm in the following way. We have a data set of which the size grows with the number of sessions and corresponding click log items. The expected sample size will be in the tens of millions of sessions. The CTR models and the Position-Based Propensity Model have a time complexity of $O(|S|)$ and $O(|S| \cdot K)$ respectively where $|S|$ denotes the total number of click log items in S and K denotes the number of iterations of the EM algorithm. The naive implementation of the algorithms would induce long computational times and is not scalable considering the expected sample sizes and the time complexity of the models. However, we can optimize the algorithms and significantly reduce the time complexity of the implemented algorithms. Specifically, we can leverage the property of the algorithms that for each unique query-document-position triple $\{(q, d, k)\}$ the estimation of the latent variables is identical for both $c = 1$ and $c = 0$. Thus, we can sum the number of clicks $c = 1$ and the number of non-clicks $c = 0$ for every $\{(q, d, k)\}$ over all click log items in S . This yields a set of query-document-position triples and corresponding click counts $\{(q, d, k, click_count, no_click_count)\}$. This set is significantly smaller compared to $|S|$. Furthermore, the size of the click log grows linearly with the number of interactions, whereas the set of query-document-position triples grows only with the number of query-document-position combinations. The dataset is now reduced from the total number of click log items $|S|$ to the number of unique query-document-position triples $|\{(q, d, k)\}|$. To outline the achieved scalability we report both the training time of the naive and optimized implementation of the EM algorithm in Chapter 7. The optimized algorithm is formalized

6. IMPLEMENTATION

in Algorithm 4. The proposed solution also applies to the MLE algorithm for the CTR models but for brevity and because of the similarity with the Expectation-Maximization algorithm we do not discuss the details.

6.3 Potential Use Cases

The initial phase of the research was not concerned with specific use cases of position bias estimation but rather with finding out how Picnic can perform position bias estimation primarily. However, during the thesis period we defined and discussed different potential use cases of the implemented position bias estimation system. In this section we will answer the following research questions:

RQ6: What are potential use cases of position bias estimation within Picnic?

[REDACTED]

Chapter 7

Results

In this chapter we present the results of the thesis. We first present and discuss the results of the experiments on the sample size and the EM iterations threshold value, as these results are used in subsequent experiments. Next, we present the results on the model performance with respect to the log-likelihood of the CTR Models and the Position-Based Propensity Model. Furthermore, we present the position bias estimated by the standard Position-Based Propensity Model. The different experiments on improvements and extensions of the Position-Based Propensity Model as well as experiments on personalized ranking are discussed next. Lastly, we discuss the results on the effectiveness of the implemented scalable Expectation-Maximization algorithm.

7.1 Sample Size

We first discuss the experiment on the sample size as the output of this experiment is used to determine the window size of the subsequent experiments. Figure 7.1 depicts the log-likelihood score of the models over the range of increasing sample sizes. We can observe that the performance of the Document-Based CTR and Position-Based Propensity Model increase with the sample size whereas the Rank-Based CTR model slightly decreases in performance. This can be explained by the fact that the DCTR and PBM have a large number of parameters compared to the RCTR that has a limited number of parameters. Intuitively, the model cannot learn more from an increased sample size. Interesting to note is the similar increase in the performance curve of the PBM and DCTR. This pattern can be explained by the fact that the parameter space of the PBM is a proper superset of the parameter space of the DCTR. Based on the result of this experiment we take a window

7. RESULTS

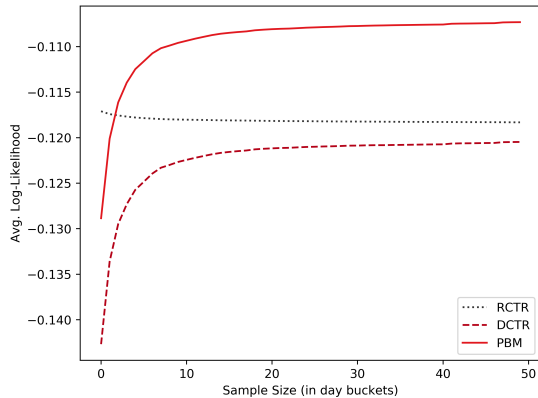


Figure 7.1: Model performance with respect to sample size (in day buckets).

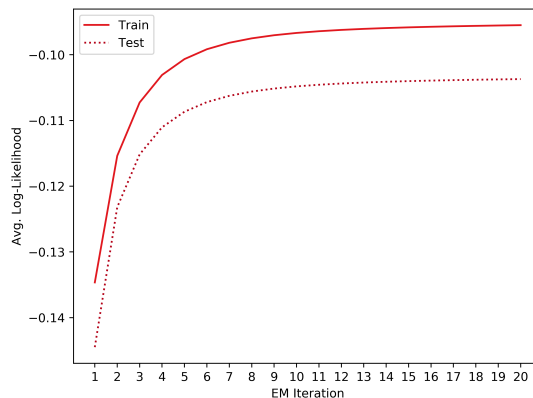


Figure 7.2: PBM EM convergence plot truncated at iteration 20.

size of 30 days, to keep the balance between sufficient natural result randomization and limited long-term relevance variations.

7.2 Expectation-Maximization Iteration Threshold

Figure 7.2 depicts the convergence plot of the Expectation-Maximization algorithm for the Position-Based Propensity Model. The plot is truncated at iteration 20. We can observe that the convergence rate is fast in the initial phase, as expected. The convergence rate slows down significantly in later iterations and is only slowly converging at iteration 20. The model is not yet fully converged at iteration 100 but the increase in log-likelihood is minimal from iteration 20. Thus, to limit computational run-time we take the value of 20 iterations for the subsequent algorithm runs. We observe similar convergence plots on all folds over the considered data set.

7.3 Model Performance

As discussed in Section 5.4, we cannot directly evaluate the position bias estimation. However, we can evaluate the fit of the model on a hold-out set of the observed data by means of the average log-likelihood score and compare the score over different models. The log-likelihood performance of the models provides an indication of the position bias estimation accuracy. Figure 7.3 depicts the arithmetic mean of the average log-likelihood scores of the RCTR, DCTR and PBM over the different windows as well as the 95% confidence intervals. Furthermore, Table 7.2 lists the log-likelihood scores over all sixteen windows. We

7. RESULTS

Table 7.1: Model performance differences, * indicates statistically significant.

	RCTR-DCTR	PBM-RCTR
Pairwise Difference Mean	0.01150	0.00825
p-value	<0.0001	<0.0001
Improvement	2.54%*	7.82%*

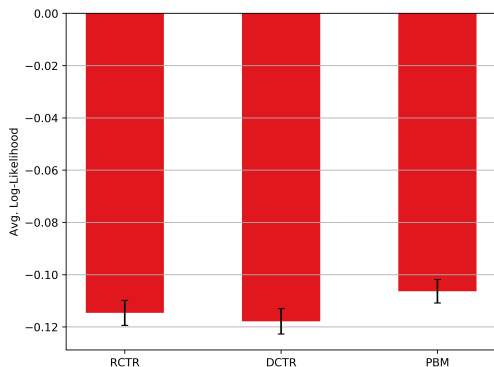


Figure 7.3: Average model performance over all folds, error bars indicate 95% CI.

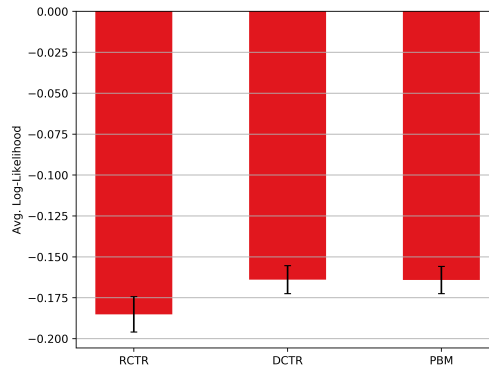


Figure 7.4: Average model performance over all folds on non-personalized data, error bars indicate 95% CI.

can observe that the Position-Based Propensity Model outperforms both baseline models. Furthermore, we observe that the Rank-Based CTR, though having a limited number of parameters, outperforms the Document-Based CTR model. A similar result is obtained by Chuklin et al. [17]. Table 7.1 lists the results of the t-test between the models and we can observe that the RCTR significantly outperforms the DCTR and the PBM significantly outperforms both baseline models. This shows that the estimated parameters of the Position-Based Propensity Model better fit the observed click data. This is a strong indication of the accuracy of the position bias estimation that is derived from the parameters of the Position-Based Propensity Model.

7.4 Position Bias Estimation

Figure 7.5 depicts the position bias as estimated by the Position-Based Propensity Model, for the last considered fold, truncated at position 20. The position bias curve for all positions is included in the Appendix 8.1. The plot depicts the estimated position bias curve as well as the empirical CTR and the display rate. The latter two rates can be considered as the lower and upper bound for the position bias respectively. As discussed in

7. RESULTS

Table 7.2: Log-likelihood score of implemented models over all folds.

Fold	RCTR	DCTR	PBM	Smoothed PBM	Noise Aware PBM
1	-0.1168	-0.1157	-0.1055	-0.1043	-0.1028
2	-0.1139	-0.1170	-0.1065	-0.1060	-0.1017
3	-0.1127	-0.1214	-0.1119	-0.1111	-0.1066
4	-0.1092	-0.1094	-0.1005	-0.0998	-0.0972
5	-0.0998	-0.1029	-0.0942	-0.0934	-0.0961
6	-0.1033	-0.1047	-0.0939	-0.0930	-0.0912
7	-0.1017	-0.1034	-0.0917	-0.0909	-0.0893
8	-0.1056	-0.1127	-0.1001	-0.0991	-0.0968
9	-0.1182	-0.1225	-0.1097	-0.1089	-0.1069
10	-0.1193	-0.1221	-0.1093	-0.1085	-0.1062
11	-0.1197	-0.1235	-0.1107	-0.1096	-0.1067
12	-0.1156	-0.1193	-0.1064	-0.1056	-0.1026
13	-0.1166	-0.1229	-0.1095	-0.1087	-0.1049
14	-0.1313	-0.1337	-0.1227	-0.1219	-0.1074
15	-0.1284	-0.1299	-0.1175	-0.1169	-0.1071
16	-0.1215	-0.1243	-0.1108	-0.1103	-0.1029
Mean	-0.1146	-0.1178	-0.1063	-0.1055	-0.1017

7. RESULTS

Section 3.5 the position bias is equivalent to the probability of examination per position. The display rate is an upper bound for the probability of examination as the customer can never examine a product that was not displayed. Likewise — since we assume that $E = 0 \implies C = 0$ — the click rate is a lower bound on the examination probability. We observe that the position bias curve correctly lies between the upper and lower bound. Furthermore, we observe that the curve is more inclined towards lower bound than the upper bound.

We observe that the propensity curve is steeper — indicating higher bias — compared to the position bias estimated for the e-commerce scenario considered by Aslanyan and Porwal [20]. We hypothesize that this is due to the specific characteristics of an online grocery store as discussed in Chapter 2. For example, as customers are familiar with the assortment the search engine might be used more for navigation than for exploration. Furthermore, this might be explained by the intuition that searches for a product in grocery stores are less deliberate compared to searches in other type of stores. Furthermore, we observe that we have less position bias compared to web search as estimated by Agarwal et al. [6]. In the extended plot in Figure 8.1 we observe that the position bias estimation is inaccurate near the lowest positions. This is most likely due to click and impression sparsity on these lower ranks. We propose to solve this by imputation of the position bias values from a certain anchor position as the differences in position bias are relatively small for the lower positions.

We can clearly observe a pagination effect in the display rate. This is due to the fact that the product tiles are displayed in a grid with two columns. As such, both positions on the same row have an equivalent display rate. Notable is that the same effect can be observed in the examination curve. We observe that the customer is more likely to examine both columns in each row of product tiles when scrolling through the result list. However, the Picnic search result page has no explicit pagination. Hence, we do not observe a pagination effect similar to other research [20].

We observe differences between the position bias estimated over the different windows. For brevity we present the position bias estimation on the latest time window, fold sixteen. We hypothesize that the differences are either due to seasonality or variation in the active customer base. As Picnic is a fast growing company, the customer base is not as static as in some other scenarios considered in the literature.

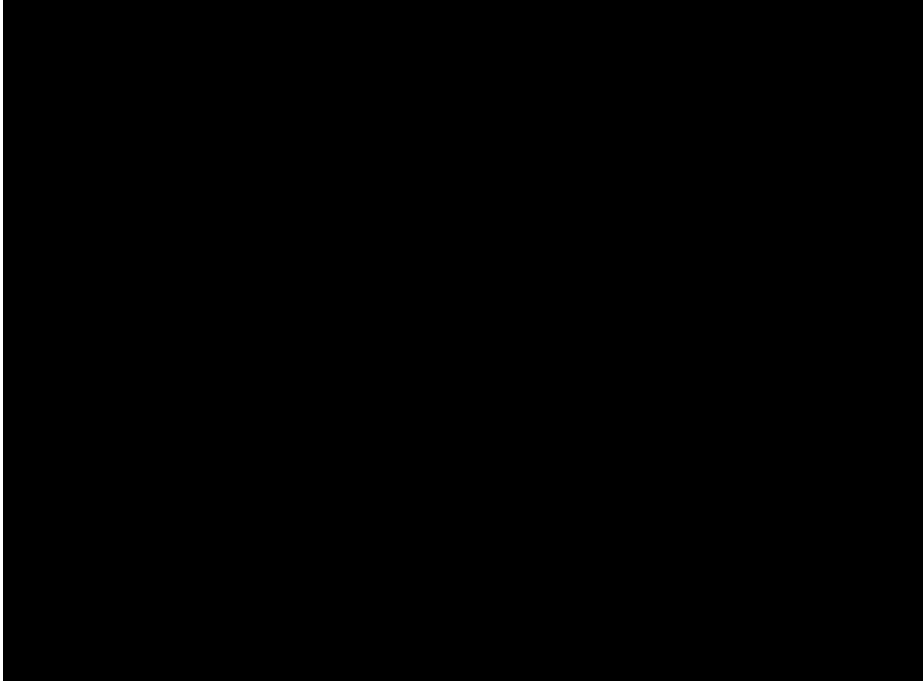


Figure 7.5: Position bias estimated by the standard Position-Based Propensity Model.

7.5 Experiments

In this section we discuss the outcome of various experiments on improvements and extensions of the Position-Based Propensity Model. The result of the experiments on the influence of personalized ranking on position bias are discussed last.

7.5.1 Expectation-Maximization Initial Values

Based on the experiments with the initial values as discussed in Section 5.5.2 we find that the algorithm converges to a global maximum with the initial values derived from the DCTR and RCTR models. The experiments prove that the initial values derived from the RCTR and DCTR model provide sufficient prior knowledge on the parameter values of the Position-Based Propensity Model for the algorithm to not get stuck in a local maximum or saddle point. The convergence plots of the experiments are included in the Appendix depicted in Figures 8.2, 8.3, 8.4 and 8.5.

7.5.2 Expectation-Maximization Data Smoothing

The log-likelihood scores of the data smoothing experiment are listed in Table 7.2. We observe a slight improvement of the log-likelihood score over the standard Position-Based

7. RESULTS

Table 7.3: Model performance differences of proposed extension, * indicates statistically significant.

	Noise Aware	Smoothing	Noise Aware + Smooth
Pairwise Difference Mean	-0.004656	-0.000806	-0.007719
p-value	0.0002	<0.0001	<0.0001
Improvement	4.33%*	0.75%*	7.25%*

Propensity Model. Table 7.3 lists the result of the significance test and shows that the smoothing extension provides a significant improvement over the standard Position-Based Propensity Model.

7.5.3 Noise Aware Position-Based Propensity Model

As discussed in Section 5.5.4, we propose a noise aware extension of the Position-Based Propensity Model that utilizes order data. Table 7.2 lists the scores over all folds of the standard and the proposed noise aware Position-Based Propensity Model. Furthermore, Table 7.3 lists the difference in model performance as well as the statistics on the significance test. We observe that the noise aware PBM significantly outperform the standard PBM. Figure 7.6 depicts the position bias curves estimated by the Position-Based Propensity Model and the noise aware extension of the PBM. We observe that the the noise aware PBM estimates a slightly higher position bias. Unlike Agarwal et al. [6], who find the opposite effect of estimating less position bias after noise modeling. We implemented a version of the Noise Aware Position-Based Propensity Model that also incorporates data smoothing. This model achieves the best performance of all models. As listed in Table 7.3, the model has a significant improvement of 7.25% over the standard Position-Based Propensity Model. This is a relatively high performance increase compared to the improvement of the PBM over the baseline models.

7.5.4 Display Events Position-Based Propensity Model

Due to the nature of the structure of the display event data in the DWH system we conducted the initial experiment on a single fold. The log-likelihood score of the models are listed in Table 7.4. The hypothesis was that the proposed extension would improve the position bias estimation as it reveals more details on user behaviour. Unexpectedly, we observe that the model performance is worse than the standard Position-Based Propensity Model. The position bias curve of both the models is depicted in Figure 7.7 and explains

7. RESULTS

Table 7.4: Log-likelihood score of PBM display events extension on fold 13.

Fold	Standard PBM	PBM Display Events
13	-0.110	-0.119



Figure 7.6: Position bias estimated by the noise aware model extension.

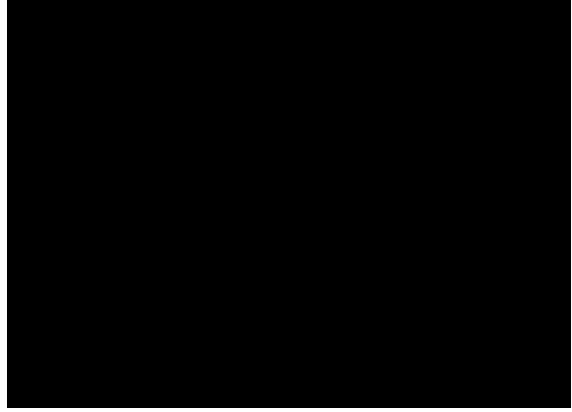


Figure 7.7: Position bias estimated by the display events model extension.

why the model performance is worse. We can observe that the position bias curve intersects with the empirical CTR curve and estimates a lower examination probability than CTR which is impossible under the assumptions of the Position-Based Propensity Model. We hypothesize that this is due to one of the following reasons:

- The proposed extension of the model could be misspecified.
- A combination of increased levels of noise in the granular display event data and a strict assumption $v = 0 \implies E = 0$.
- Imbalance of display rates between the top four position and the other positions.

7.5.5 Personalized Ranking

To determine the influence of personalized ranking on position bias, we conducted two experiments as discussed in Chapter 5.

Figure 7.4 depicts the model scores on the filter based personalized experiments. We observe two notable differences compared to the complete data set results. First, we observe that for the filter based approach the overall model performance is worse. This can be attributed to either the significantly reduced sample size due to filtering or the fact

7. RESULTS

that the models that are trained on the complete data set have an easier task of predicting patterns that emerge more often in the training set.

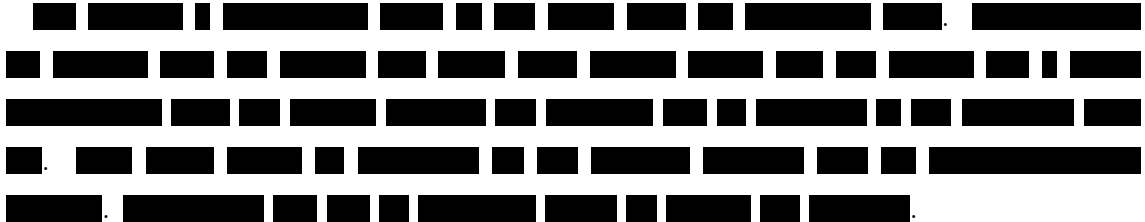
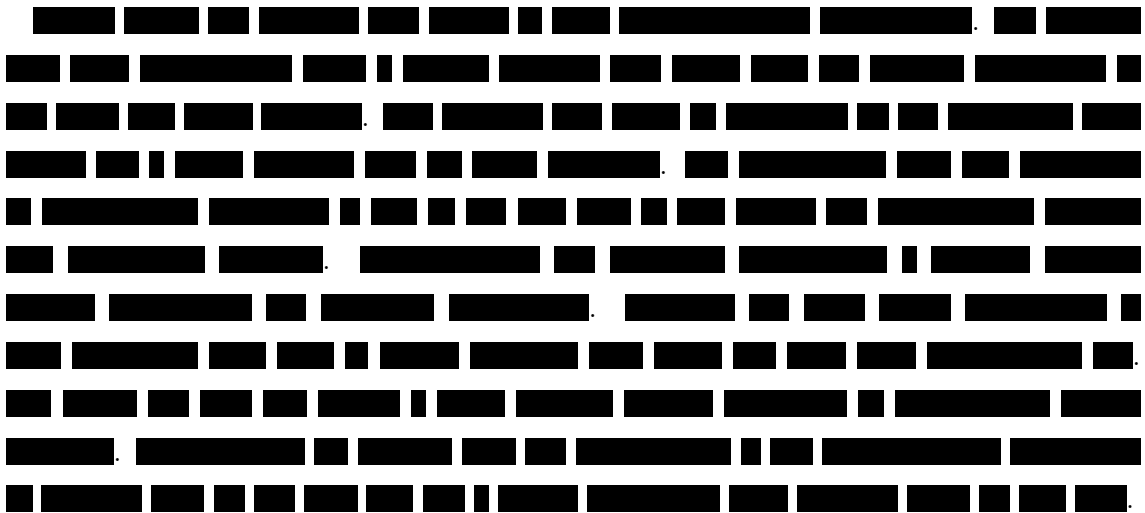


Figure 7.8 depicts the position bias curve estimated from the non-personalized ranking experiment as well as the empirical CTR and position bias as estimated on the complete data. We observe both a higher empirical CTR and less position bias on the non-personalized rankings. Although we cannot verify the assumptions, we hypothesize that this is related to quality-of-context bias [7, 9] as the quality of the overall ranking also influences the user interactions. Customers might examine more documents in a non-personalized ranking due to a lower overall quality of the search ranking. A personalized ranking will most likely increase overall quality of the ranking and thus decrease examination of lower ranks, consequently increasing position bias. Notable, is that both curves follow similar pairwise patterns on the lower ranked position.



7.6 Scalability

Table 7.5 lists statistics on the size of the data set as well as the training time of the Position-Based Propensity Model EM algorithm for each fold. We observe that the average training time of the PBM EM algorithm is 100.6 seconds. Furthermore, We can observe that the number of (q, d, k) triples is significantly lower compared to the number of click

7. RESULTS

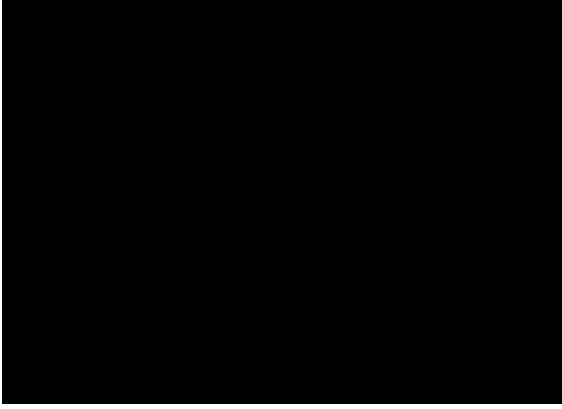


Figure 7.8: Estimated position bias for the filter based personalization experiment.

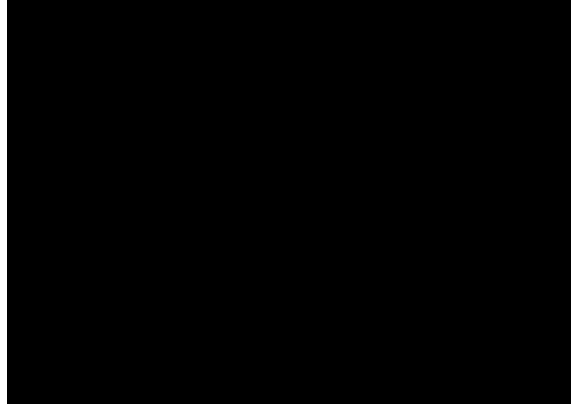


Figure 7.9: Estimated position bias for both personalization experiments.

log items for each fold. The average number of click log items is over 99 million whereas the average number of (q, d, k) triples is just over a million. This indicates that the proposed optimization of the algorithm can ensure scalability. From the number of click log items we can obtain the potential runtime of the system had it not been optimized. $\frac{99,228,196}{1,060,636} \times 100.6 \approx 9,411.67$ seconds. Thus, the non-optimized algorithm would have a potential runtime of more than two hours. We conclude that the implemented optimized version of the algorithm provides sufficient run time performance for the current expected data size. Future growth of the data size is expected as the number of customer, products and consequently queries will increase over time. However, the system will still be able to handle such data volumes. Even if the number of (q, d, k) will grow by a ten fold the system will still have a reasonable run-time.

7. RESULTS

Table 7.5: Data size statistics and PBM training time (in seconds) over all folds.

Fold	Sessions	Click log items	(q,d,k)	Queries	PBM training time
1	2,915,116	76,728,027	756,922	3238	73.5
2	2,763,679	77,299,496	796,319	3220	80
3	3,089,353	89,337,729	886,121	3413	85
4	2,772,175	82,642,911	1,038,857	3315	100
5	3,000,112	92,743,723	1,007,071	3503	98.5
6	2,931,115	98,327,899	1,782,891	3830	168
7	2,483,803	79,004,589	1,156,910	3493	112
8	2,728,016	86,516,685	1,118,815	3492	106
9	3,317,873	98,819,145	1,216,776	3835	115.5
10	3,526,928	92,548,748	1,064,483	4074	102
11	3,764,510	97,787,309	1,040,645	4227	98
12	3,726,942	95,815,469	946,772	4275	89
13	4,402,104	114,463,943	1,009,862	4517	92.5
14	4,434,101	117,199,812	963,473	4522	88
15	5,597,963	148,715,459	1,116,416	5016	102
16	5,450,480	139,700,194	1,067,841	5260	99.5
Avg.	3,556,517	99,228,196	1,060,636	3952	100.6

Chapter 8

Conclusion

In this thesis we set out to answer the following research question:

How can we develop and implement a system to estimate position bias for unbiased learning-to-rank in the context of Picnic?

In conclusion, we propose and implement a novel system for position bias estimation in the context of an online grocery store or similar e-commerce stores. We employ the Position-Based Propensity Model to estimate position bias utilizing add to basket events. The model parameters are learned through a scalable implementation of the Expectation-Maximization algorithm. We put forward a production ready system with the joint objective of position bias estimation and pointwise relevance learning-to-rank for head queries. Furthermore, we put forward a Python implementation of the Rank-Based CTR, Document-Based CTR and Position-Based Propensity Model. We find that the implemented Position-Based Propensity Model significantly outperforms both baseline CTR models. Furthermore, we find that the proposed noise aware extension of the Position-Based Propensity Model presents a significant improvement over the standard model. We experimented with an extension of the Position-Based Propensity Model that incorporates display event data. However, based on our experiments we find that this extension cannot improve the model performance.

8.1 Future Work

In this section we discuss the limitations and future work of the thesis. We acknowledge the limitation of the research in the fact that we are not able to verify the propensity estimations by learning an unbiased learning-to-rank model. The performance of the unbiased

8. CONCLUSION

and biased learning-to-rank models can be compared to determine if the proposed propensity estimation is accurate. Furthermore, the experiments on the display event extension of the Position-Based Propensity Model are limited due to time constraints of the thesis. We acknowledge that we cannot preclude that the proposed model can improve performance under certain conditions of noise-free and balanced display event data.

Furthermore, as discussed in Section 3.2, user interaction data can be affected by other types of biases such as presentation bias and selection bias. We acknowledge that both biases can also affect user interactions in the Picnic data. The effect of selection bias is mostly a future problem as the current assortment is limited. Presentation bias could already have a significant effect on the Picnic user interaction data. We leave the estimation of both bias effects for future work.

Furthermore, the search ranking layout in the Picnic app is a grid-based layout. Oosterhuis and de Rijke [51], Xie et al. [52] prove that more complex ranking layouts such as grid-based layouts can introduce additional biases such as middle-bias. However, as the number of columns is limited the user interaction are not affected by middle bias. Furthermore, these additional biases are not considered in the unbiased learning-to-rank framework yet. We leave it for future work to also consider these additional biases.

Moreover, due to the inherent limitations of the standard Position-Based Propensity Model we cannot directly improve the model using additional features. Consequently, we cannot conclude that the implemented model accounts for all unobserved covariates in terms of product relevance. For example, variations in the price due to promotions and discounts might influence product relevance. These covariates are not directly incorporated in the model. However, the observed performance of the implemented models does not indicate that this poses a point of concern.

8.2 Recommendations

In addition to the recommendations on the use cases, as discussed in Section 6.3. We have the following recommendation for Picnic. Employing the estimated click propensities Picnic can obtain an invaluable quantity of unbiased user interaction data. We recommend Picnic to utilize this data and develop an unbiased learning-to-rank model for search and potentially other rankings in the application. Current advances in both the literature and the practice indicate that this is a worthwhile investment as discussed in the thesis. An extension of the well establish machine learning platform TensorFlow incorporates support for unbiased learning-to-rank [53].

8. CONCLUSION

Regarding other biases, we recommend to first start with development of a learning-to-rank model before further exploring how other biases affect the user interaction data of Picnic. Furthermore, we recommend to verify the accuracy of the estimated propensity of the proposed system by means of A/B testing both an biased and unbiased learning-to-rank model. This recommendation also applies to context aware position bias and customer specific position bias.

Lastly, we recommend to impute the propensity values for the lowest ranks by the propensity of a preceding anchor position. Due to data sparsity these values are most likely to be incorrect.

References

- [1] Hang Li. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011. 1, 11, 41
- [2] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 115–124, 2016. 1, 4, 12, 18, 20, 23, 30, 33, 38, 41
- [3] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017. 1, 12, 16, 18, 19, 20, 23, 24, 41
- [4] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. To model or to intervene: A comparison of counterfactual and online learning to rank from user interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 15–24, 2019. 1, 38
- [5] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–484, 2017. 1, 2
- [6] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 474–482, 2019. 1, 2, 4, 15, 20, 27, 28, 31, 41, 53, 55
- [7] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query

REFERENCES

- reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7–es, 2007. 1, 12, 57
- [8] Yisong Yue, Rajan Patel, and Hein Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, page 1011–1018, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605587998. doi: 10.1145/1772690.1772793. URL <https://doi.org/10.1145/1772690.1772793>. 1, 12
- [9] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. *SIGIR Forum*, 51(1):4–11, August 2017. ISSN 0163-5840. doi: 10.1145/3130332.3130334. URL <https://doi.org/10.1145/3130332.3130334>. 1, 12, 41, 57
- [10] Aman Agarwal, Xuanhui Wang, Cheng Li, Mike Bendersky, and Marc Najork. Addressing trust bias for unbiased learning-to-rank. In *Proceedings of the 2019 World Wide Web Conference*, pages 4–14, 2019. 1, 12, 38, 41
- [11] Maeve O’Brien and Mark T Keane. Modeling result-list searching in the world wide web: The role of relevance topologies and trust bias. In *Proceedings of the 28th annual conference of the cognitive science society*, volume 28, pages 1881–1886. Citeseer, 2006. 1, 12
- [12] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. Learning from user interactions in personal search via attribute parameterization. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 791–799, 2017. 1
- [13] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 610–618, 2018. 1, 4, 12, 19, 20, 23, 24, 25, 26, 30, 31, 34, 37, 39
- [14] Judit Bar-Ilan, Kevin Keenoy, Mark Levene, and Eti Yaari. Presentation bias is significant in determining user preference for search results—a user study. *Journal of the American Society for Information Science and Technology*, 60(1):135–149, 2009. 1, 12

REFERENCES

- [15] Anjan Goswami, ChengXiang Zhai, and Prasant Mohapatra. Towards optimization of e-commerce search and discovery. In *eCOM@ SIGIR*, 2018. 2
- [16] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94, 2008. 2, 15, 16
- [17] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services*, 7(3):1–115, 2015. 2, 14, 15, 16, 20, 25, 32, 33, 34, 35, 37, 38, 39, 40, 51
- [18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 2, 14, 25, 40
- [19] Yinxiao Li. Handling position bias for unbiased learning to rank in hotels search. *arXiv preprint arXiv:2002.12528*, 2020. 4, 20, 38
- [20] Grigor Aslanyan and Utkarsh Porwal. Position bias estimation for unbiased learning-to-rank in ecommerce search. In *International Symposium on String Processing and Information Retrieval*, pages 47–64. Springer, 2019. 4, 24, 28, 29, 31, 38, 40, 53
- [21] Stefan Büttcher, Charles LA Clarke, and Gordon V Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016. 8, 9
- [22] Hang Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011. 11, 12
- [23] Gai Li and Qiang Chen. Exploiting explicit and implicit feedback for personalized ranking. *Mathematical Problems in Engineering*, 2016, 2016. 11
- [24] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. Unbiased lambdamart: An unbiased pairwise learning-to-rank algorithm. In *The World Wide Web Conference*, pages 2830–2836, 2019. 12
- [25] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 13, 40
- [26] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530, 2007. 15

REFERENCES

- [27] 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*, pages 331–338, 2008. 15, 16
- [28] Fan Guo, Chao Liu, and Yi Min Wang. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM '09*, page 124–131, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605583907. doi: 10.1145/1498759.1498818. URL <https://doi.org/10.1145/1498759.1498818>. 15, 40
- [29] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web*, pages 11–20, 2009. 15, 37
- [30] Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10, 2009. 15, 16, 25, 40, 41
- [31] Jeff Huang, Ryen W White, and Susan Dumais. No clicks, no problem: using cursor movements to understand and improve search. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1225–1234, 2011. 15
- [32] Jeff Huang, Ryen W White, Georg Buscher, and Kuansan Wang. Improving searcher models using mouse cursor activity. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 195–204, 2012. 15, 37, 41
- [33] Andrew Turpin, Falk Scholer, Kalvero Jarvelin, Mingfang Wu, and J. Shane Culpeper. Including summaries in system evaluation. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09*, page 508–515, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584836. doi: 10.1145/1571941.1572029. URL <https://doi.org/10.1145/1571941.1572029>. 15
- [34] Weizhu Chen, Dong Wang, Yuchen Zhang, Zheng Chen, Adish Singla, and Qiang Yang. A noise-aware click model for web search. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 313–322, 2012. 15, 41

REFERENCES

- [35] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 5–14, 2019. 18
- [36] Artem Grotov, Aleksandr Chuklin, Ilya Markov, Luka Stout, Finde Xumara, and Maarten de Rijke. A comparative study of click models for web search. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 78–90. Springer, 2015. 20
- [37] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. Intervention harvesting for context-dependent examination-bias estimation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 825–834, 2019. 20, 41
- [38] Aman Agarwal, Ivan Zaitsev, and Thorsten Joachims. Consistent position bias estimation without online interventions for learning-to-rank, 2018. 20, 27
- [39] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. Cascade model-based propensity estimation for counterfactual learning to rank. *arXiv preprint arXiv:2005.11938*, 2020. 20
- [40] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 385–394, 2018. 28, 31
- [41] Revlin Abbi, Elia El-Darzi, Christos Vasilakis, and Peter Millard. Analysis of stopping criteria for the em algorithm in the context of patient grouping according to length of stay. In *2008 4th International IEEE Conference Intelligent Systems*, volume 1, pages 3–9. IEEE, 2008. 35
- [42] Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. A novel click model and its applications to online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 321–330, 2010. 37

REFERENCES

- [43] Christoph Bergmeir, Rob J Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, 2018. 38
- [44] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020. 38
- [45] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, 2019. 38
- [46] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. 38
- [47] Ilya Markov, Alexey Borisov, and Maarten de Rijke. Online expectation-maximization for click models. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2195–2198, 2017. 38
- [48] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012. 39
- [49] CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983. 40
- [50] Yiqun Liu, Chao Wang, Ke Zhou, Jianyun Nie, Min Zhang, and Shaoping Ma. From skimming to reading: A two-stage examination model for web search. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 849–858, 2014. 41
- [51] Harrie Oosterhuis and Maarten de Rijke. Ranking for relevance and display preferences in complex presentation layouts. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR ’18*, page 845–854, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3209992. URL <https://doi.org/10.1145/3209978.3209992>. 61
- [52] Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Maarten de Rijke, Yunqiu Shao, Zixin Ye, Min Zhang, and Shaoping Ma. Grid-based evaluation metrics for web image search. In *The World Wide Web Conference, WWW ’19*, page 2103–2114, New York, NY, USA,

REFERENCES

2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313514. URL <https://doi.org/10.1145/3308558.3313514>. 61
- [53] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2019. doi: 10.1145/3292500.3330677. URL <http://dx.doi.org/10.1145/3292500.3330677>. 61

Appendix

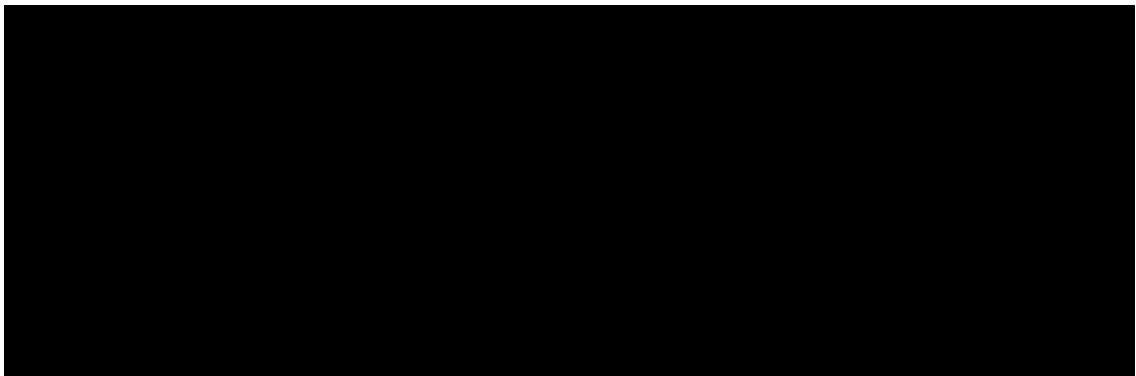


Figure 8.1: Position bias estimated by the standard Position-Based Propensity Mode for all positions.

Algorithm 2: Display Events PBM Expectation-Maximization

```

input :  $S_v, \{\rho_k\}, \{\rho_{q,d}\}$ , iterations
output:  $\{\theta_k\}, \{\gamma_{q,d}\}$ 
1 Initialize Parameters
2  $\{\theta_k\} \leftarrow \{\rho_k\}$ 
3  $\{\gamma_{q,d}\} \leftarrow \{\rho_{q,d}\}$ 
4  $t \leftarrow 0$ 
5 while  $t < \text{iterations}$  do
6      $\text{ESS}_\theta[\text{sum}_k] \leftarrow 0$ 
7      $\text{ESS}_\gamma[\text{sum}_{q,d}] \leftarrow 0$ 
8     foreach  $s \in S$  do
9         foreach  $(q, d, k, c, v) \in s$  do
10            if  $c = 1$  then
11                 $E = R = 1$  as per 5.3
12            else
13                if  $v=1$  then
14                     $E = 0$ 
15                     $R = P(E=0, R=1 | C=0, q, d, k)$  estimate using 5.13
16                else
17                     $E = P(E=1, R=0 | C=0, q, d, k)$  estimate using 5.4
18                     $R = P(E=0, R=1 | C=0, q, d, k)$  estimate using 5.5
19                end
20            end
21             $\text{ESS}_\theta[\text{sum}_k] += E$ 
22             $\text{ESS}_\gamma[\text{sum}_{q,d}] += R$ 
23        end
24    end
25    foreach  $\theta_k \in \{\theta_k\}$  do
26         $\theta_k = \frac{1}{|S_k|} \cdot \text{ESS}_\theta[\text{sum}_k]$ 
27    end
28    foreach  $\gamma_{q,d} \in \{\gamma_{q,d}\}$  do
29         $\gamma_{q,d} = \frac{1}{|S_{q,d}|} \cdot \text{ESS}_\gamma[\text{sum}_{q,d}]$ 
30    end
31     $t = t + 1$ 
32 end

```

Algorithm 3: Noise Aware PBM Expectation-Maximization

```

input :  $S_b, \{\rho_k\}, \{\rho_{q,d}\}$ , iterations
output:  $\{\theta_k\}, \{\gamma_{q,d}\}$ 
1 Initialize Parameters
2  $\{\theta_k\} \leftarrow \{\rho_k\}$ 
3  $\{\gamma_{q,d}\} \leftarrow \{\rho_{q,d}\}$ 
4  $t \leftarrow 0$ 
5 while  $t < \text{iterations}$  do
6   Expectation Step
7   Estimate distribution of latent parameters E and R for every click log item
8   Calculate ESS for each parameter based on estimated distributions.
9    $\text{ESS}_\theta[\text{sum}_k] \leftarrow 0$ 
10   $\text{ESS}_\gamma[\text{sum}_{q,d}] \leftarrow 0$ 
11  foreach  $s \in S$  do
12    foreach  $(q, d, k, c, b) \in s$  do
13      if  $c = 1$  and  $b = 1$  then
14         $E = R = 1$  as per 5.3
15      else
16         $E = P(E=1, R=0 | C=0, q, d, k)$  estimate using 5.4
17         $R = P(E=0, R=1 | C=0, q, d, k)$  estimate using 5.5
18      end
19       $\text{ESS}_\theta[\text{sum}_k] += E$ 
20       $\text{ESS}_\gamma[\text{sum}_{q,d}] += R$ 
21    end
22  end
23  Maximization Step
24  foreach  $\theta_k \in \{\theta_k\}$  do
25     $\theta_k = \frac{1}{|S_k|} \cdot \text{ESS}_\theta[\text{sum}_k]$ 
26  end
27  foreach  $\gamma_{q,d} \in \{\gamma_{q,d}\}$  do
28     $\gamma_{q,d} = \frac{1}{|S_{q,d}|} \cdot \text{ESS}_\gamma[\text{sum}_{q,d}]$ 
29  end
30   $t = t + 1$ 
31 end

```

Algorithm 4: Scalable PBM Expectation-Maximization

```

input :  $S, \{\rho_k\}, \{\rho_{q,d}\}, \text{iterations}$ 
output:  $\{\theta_k\}, \{\gamma_{q,d}\}$ 
1 Initialize Parameters
2  $\{\theta_k\} \leftarrow \{\rho_k\}$ 
3  $\{\gamma_{q,d}\} \leftarrow \{\rho_{q,d}\}$ 
4  $t \leftarrow 0$ 
5 while  $t < \text{iterations}$  do
6   Expectation Step
7   Estimate distribution of latent parameters E and R for every click log item
8   Calculate ESS for each parameter based on estimated distributions.
9    $\text{ESS}_\theta[\text{sum}_k, \text{count}_k] \leftarrow 0$ 
10   $\text{ESS}_\gamma[\text{sum}_{q,d}, \text{count}_{q,d}] \leftarrow 0$ 
11  foreach  $(q,d,k,c\_sum,nc\_sum) \in s$  do
12    if  $c = 1$  then
13      |  $E = R = c\_sum$  as per 5.3
14    else
15      |  $E = (P(E=1,R=0|C=0,q,d,k) \cdot n\_sum)$  estimate using 5.4
16      |  $R = (P(E=0,R=1|C=0,q,d,k) \cdot n\_sum)$  estimate using 5.5
17    end
18     $\text{ESS}_\theta[\text{sum}_k] += E$ 
19     $\text{ESS}_\gamma[\text{sum}_{q,d}] += R$ 
20  end
21  Maximization Step
22  foreach  $\theta_k \in \{\theta_k\}$  do
23    |  $\theta_k = \frac{1}{[\text{count}_k]} \cdot \text{ESS}_\theta[\text{sum}_k]$ 
24  end
25  foreach  $\gamma_{q,d} \in \{\gamma_{q,d}\}$  do
26    |  $\gamma_{q,d} = \frac{1}{[\text{count}_{q,d}]} \cdot \text{ESS}_\gamma[\text{sum}_{q,d}]$ 
27  end
28   $t = t + 1$ 
29 end

```

Appendix

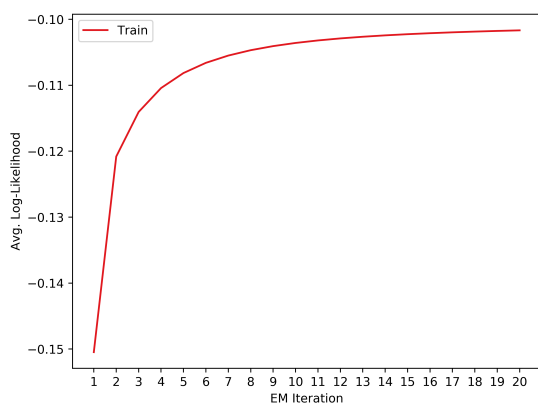


Figure 8.2: Convergence plot initial values experiment $\{\theta_k\}, \{\gamma_{q,d}\} = 0.2$.

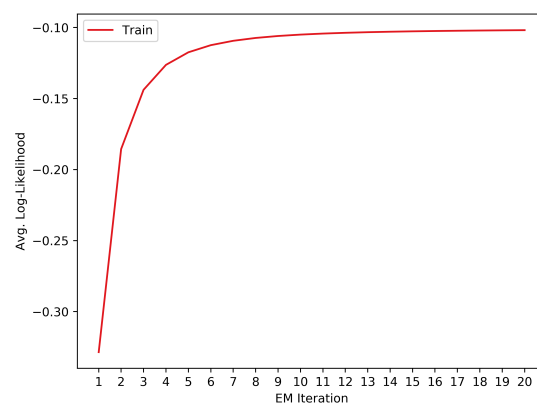


Figure 8.3: Convergence plot initial values experiment $\{\theta_k\}, \{\gamma_{q,d}\} = 0.5$.

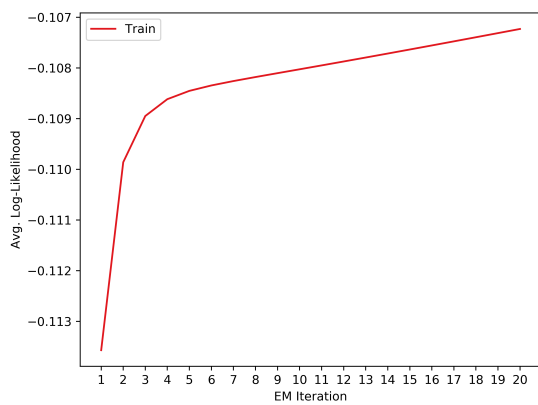


Figure 8.4: Convergence plot initial values experiment 5.10.

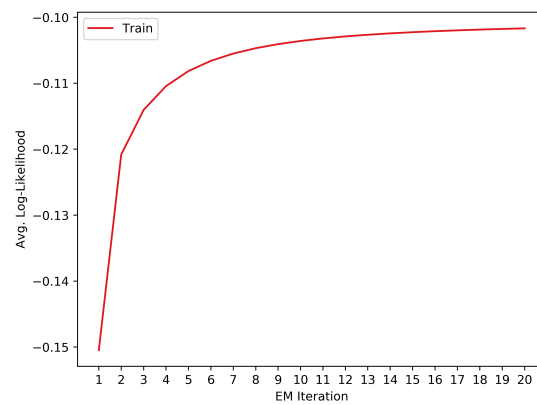


Figure 8.5: Convergence plot initial values experiment 5.11.