Modelling and Detecting Changes in User Satisfaction

Julia Kiseleva[‡,*], Eric Crestan[†], Riccardo Brigo[†], Roland Dittel[†]
[‡]Computer Science Department, Eindhoven University of Technology, Eindhoven, The Netherlands
[†]Microsoft Bing, Munich, Germany
[‡]j.kiseleva@tue.nl [†]{ericres|rbrigo|Roland.Dittel}@microsoft.com

ABSTRACT

Informational needs behind queries, that people issue to search engines, are inherently sensitive to external factors such as breaking news, new models of devices, or seasonal changes as *'black Friday'*. Mostly these changes happen suddenly and it is natural to suppose that they may cause a shift in user satisfaction with presented old search results and push users to reformulate their queries. For instance, if users issued the query *'CIKM conference'* in 2013 they were satisfied with results referring to the page `cikm2013.org` and this page gets a majority of clicks. However, the confernce site has been changed and the same query issued in 2014 should be linked to the different page `cikm2014.fudan.edu.cn`. If the link to the fresh page is not among the retrieved results then users will reformulate the query to find desired information.

In this paper, we examine how to detect changes in user satisfaction if some events affect user information goals but search results remained the same. We formulate a problem using concept drift detection techniques. The proposed method works in an unsupervised manner, we do not rely on any labelling. We report results of a large scale evaluation over real user interactions, that are collected by a commercial search engine within six months. The final datasets consist of more than sixty millions log entries. The results of our experiments demonstrate that by using our method we can accurately detect changes in user behavior. The detected drifts can be used to enhance query auto-completion, user satisfaction metrics, and recency ranking.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Information Filtering

[*]Research was performed while the author was at Microsoft Bing.

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661960.

General Terms

Algorithms, Performance, Experimentation

Keywords

Query reformulation, concept drift, information retrieval

1. INTRODUCTION

Millions of users interact with search engines daily to obtain *fresh* information quickly while minimazing their effort. Users issue a query Q and a search engine returns search result page ($SERP$) that is a ranked list of URLs: $SERP = (url_1, \ldots, url_i, \ldots, url_n)$. The order of URLs in $SERP$ is optimized to fit a history of user interactions with a pair $\langle Q, SERP \rangle$ [3]. However, events from the outside world and time can affect user behavior on the Web [24], [25]. To illustrate this drift in the user information goals let us consider the following examples:

- The last Olympics games occur in 2014, so users are not interested in the previous 2012 Olympics anymore. Therefore, if users issue the query *'Olympics games'* in 2014 they need to find a page of the latest event. If the desired link is not among the retrieved results then user satisfaction with the served $SERP$ decreases.
- After Microsoft releases a new *'Windows phone 8'* users are not satisfied if pages of previous models are in the top of $SERP$.
- After a cartoon 'Despicable Me 2' is releazed the audience pays less attention to its previous releaze.

User satisfaction with a pair $\langle Q, SERP \rangle$ can decrease dramatically if user information needs change due to some event or decay/change of interest over time. In this paper, we answer the question *'How can we detect a drift in user satisfaction with the pair $\langle Q, SERP \rangle$ using users' interactions on the $SERP$?'*

When users struggle to find an answer for Q they run a follow-up query Q' that is an expansion of Q. *Query reformulation* is the act of submitting a next query Q' to modify a previous $SERP$ for a query Q in the hope of retrieving better results [13]. Such a query reformulation is a strong indication of user dissatisfaction [1]. We call this the *reformulation signal*. Our hypothesis is that a decrease in user satisfaction with $\langle Q, SERP \rangle$ correlates nicely with the reformulation signal. In other words, the probability of reformulating Q will grow dramatically.

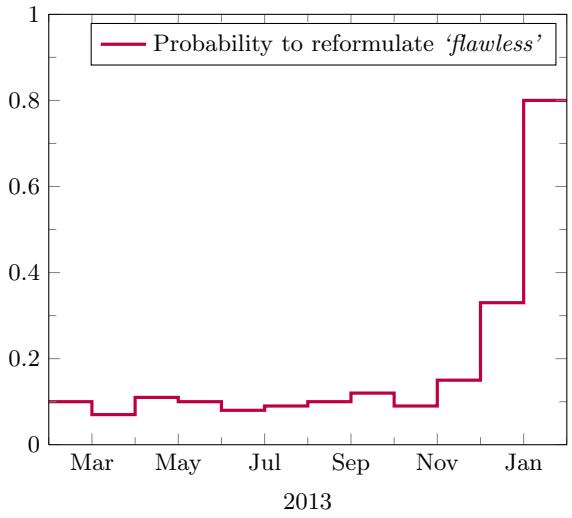

Figure 1: The histogram of the probability to reformulate query *'flawless'* in 2013 with one month granularity.

Let us consider the probability of reformulating a query *'flawless'* during year 2013. A histogram of this probability is shown in Figure 1. We can clearly see that a drift happened in December. When users ran this query before October 2013 they most probably were looking for a movie, called *'flawless'*. However, the singer Beyonce released her new soundtrack also called *'flawless'* in November 2013. Hence, this event affected dramatically the meaning of this query. As a result, if the desired song was missing in $SERP$ a majority of users reformulated the query by expanding it with the term *'Beyonce'*.

We propose an unsupervised approach for detecting drifts in user satisfaction for pairs $\langle Q, SERP \rangle$ by applying a concept drift technique [33] leveraging reformulation signal. Concept drift primarily refers to an online supervised learning scenario when the relation between the input data and the target variable changes over time [10]. Furthermore, the reformulation signal is considered to be less noisy and if reformulations are fresh and done only by users' initiative then we can say that a reformulation signal is not biased by information coming from the search engine. Moreover, the proposed method produces:

- A list of *drift terms*, that users apply to reformulate queries when a drift happens. This list can be utilised for time sensitive query auto-completion [29].
- A list of $URLs$, that users mostly click on after reformulating initial queries. This list can be used for a recency ranking [7], [18], [19].

The specific contributions of this paper include:

1. A definition of query reformulation signal as an effective way to detect an alteration in user satisfaction.
2. An analysis and formulation of a drift detection in user satisfaction.
3. An unsupervised method for detection changes in user satisfaction.
4. A large scale evaluation over real user queries, showing a high accuracy of proposed method.

The remainder of this paper is structured as follows. Section 2 describes background and related work. A formal description of the proposed method to detect changes in user satisfaction is presented in Section 3. Section 4 describes a research methodology for a large scale exploratory analysis of real user behavior logs from a commercial search engine. In Section 5 we describe obtained results. In Section 6 we discuss potential applications, that can benefit within proposed method, are described. We summarize our findings, discuss possible extensions of the current work and conclude in Section 7.

2. BACKGROUND AND RELATED WORK

Our work examines how to model and detect changes in user satisfaction that can be a useful feature for a dynamic ranking. Huffman and Hochster [16] observed a strong correlation between the relevance of results and user satisfaction using navigational and non-navigational queries. Relevance is a complex concept (for a detailed review see [26], [27]). In a simplified view relevance Rel can be defined as a score for a pair of query Q and document D, where D in this case is a link URL to the web page: $Rel{=}\|\langle Q, SERP \rangle\|$. However, it is logical to assume that Rel have an altering nature because user preferences change due to external events and passage of time. Dong et al. [7] proposed a classifier to detect recency sensitive queries. This classifier gives a score, called *'buzzines'*, to a query Q and Q is considered as a breaking-news one if its final buzzines score exceeds some threshold. Moreover, recency ranking is proposed to overcome an issue with ranking time-sensitive queries. It proposes Rel that takes a *freshness* of a document into account. Dong et al. [7] proposed to incorporate recency features in a ranking model. The ranking function includes recency features: *(timestamp, linktime, WebBuzz, page topic)* and it gives a gain for ranking metrics. The paper [18] suggests a temporal click feature, called *ClickBuzz*, that captures a spiking interest in the pair $\langle Q, SERP \rangle$. This method helps to exploit user feedback for time-sensitive queries. The use of *ClickBuzz* in the ranking models leads to an improvement in $NDCG_5$. Our method can be considered as a supplement to recency ranking, it detects moments when drifts happen and we need to adjust our ranking function in order to produce up-to-date results.

User satisfaction has been researched extensively. User clicks are reasonably accurate on average to evaluate user satisfaction with pairs $\langle Q, SERP \rangle$ [2], [22], using click-through information. This user satisfaction scenario is successfully applied to navigational queries. It is called *query-level satisfaction*. However, we have to take into account the fact that user clicks are biased:

1. to the page position in $SERP$ [6],[21];
2. to the quality of the page's snippet [32], and **(3)** to the domain of the returned URL [17].

Authors of the paper [4] claim that a search scenario for informational queries is different. Users can run follow-up queries if they are unsatisfied with the derived results. Reformulations can lead users to an answer. This scenario is called *task-level satisfaction* [7]. Past research proposed different methods for identifying successful sessions. Hassan

et al. [12] used a Markov model to predict success at the end of the task. Ageev et al. [1] exploited an expertise-dependent difference in search behavior by using a Conditional Random Fields model to predict a search success. On the other hand, separate researches are interested in situations when users are frustrated. Feild et. al [9] proposed a method for understanding user frustration with the pair $\langle Q, SERP \rangle$. Authors gave users difficult information seeking assignments and evaluated their level of dissatisfaction via query log features and physical sensors. The authors demonstrated that the prediction model gets the highest quality when it is built based on query log features, described in the paper [30]. One type of user behavior that can be clearly associated with frustration is search engine switching. Authors of the paper [11] showed that one of the primary reasons users switched their search engine was due to dissatisfaction with the results on the $SERP$.

In our work, we consider a scenario when user satisfaction at time t_i with $\langle Q, SERP \rangle$ turn into user frustration at t_{i+1} with the same $\langle Q, SERP \rangle$. We associate user satisfaction using the reformulation signal. If the probability of reformulating query Q was close to zero at t_i and grows dramatically at t_{i+1}, then a change happened in user satisfaction. Our scenario corresponds perfectly to a definition of *real concept drift*. In dynamically changing and non-stationary environments, the data distribution can change over time because of the phenomenon of *concept drift* [28], [31]. The *real* concept drift refers to changes in the conditional distribution of the output (i.e., target variable) given the input (input features), while the distribution of the input may stay unchanged. Formally *concept drift* between time point t_i and time point t_{i+1} can be defined as:

$$\exists X : p_{t_i}(X, y) \neq p_{t_{i+1}}(X, y), \quad (1)$$

where p_{t_i} denotes the joint distribution at time t_i between the set of input variables X and the target variable y. In this work, we follow a lead of [5], [8], [14] and reuse methods from supervised machine learning and statistical learning theory to design and analyze suitable statistics for drift detection.

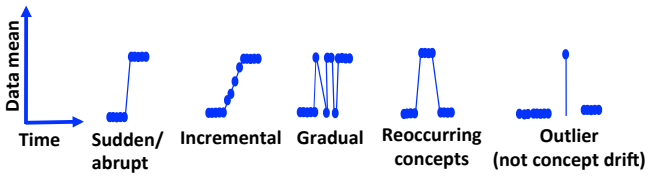

Figure 2: Patterns of changes over time [10]

Changes in data distribution over time may manifest in different forms, as illustrated in Figure 2. The presented types of concept drift are perfectly align to the reformulation signal:

- A drift may happen *suddenly/abruptly* by switching from one concept to another, that may corresponds to breaking-news queries such as *'nelson mandela'* (issued the day of his death).
- A drift can be *incremental*, e.g. a query *'cikm conference'* may drift each year, queries referring to a new model of a device may cause an incremental drift: users incrementally move their preferences from *'windows phone 7'* to *'windows phone 8'*.
- A drift can be *gradual*, e.g. relevant new topics change from *dwelling to holiday homes*, while the user does not switch abruptly, but rather keeps going back to the previous interest for some time.
- A drift can be *reoccurring*, e.g. seasonal queries: '30 % cvs coupon'→'30 % cvs coupon **black friday**'.
- One of the challenges for concept drift handling algorithms is not to mix the true drift with an outlier or noise which refers to a once-off random deviation or an anomaly. In our case, it may be spam queries. We will show an example of an outlier in Section 5.4.

To summarize, the key distinctions of our work compared to previous efforts are: a clear and well-defined approach to detecting changes in user satisfaction using the reformulation signal; an in-depth analysis of changes is searchers behavior that results in accurate detection of drifts in user satisfaction. Moreover, our framework works in an unsupervised manner, it does not require any labelling.

3. DETECTING CHANGES IN USER SATISFACTION

In this Section we present an overview of a developed framework for detecting changes in user satisfaction due some external events and overtime decay. Our framework uses the growth of the reformulation signal as an indication of user dissatisfaction with the pair $\langle Q, SERP \rangle$. In other words, if the probability to reformulate a query Q to Q' grows dramatically then users are no longer satisfied with the pair $\langle Q, SERP \rangle$. The desired results for the query Q has been changed and now users expect to derive $SERP'$ as the answer for Q.

The proposed framework monitors user interactions and it triggers an alarm to the system when changes happen. Moreover, if indicted, our framework can produce the following additional output per a query:

- a list of drift terms, which users added to reformulate the query Q;
- a list of drift $URLs$, which users clicked on after issuing Q'.

A detailed diagram of our framework is presented in Figure 3. In following sections we will describe our framework in details:

1. how do we construct *user behavioral logs* (Section 3.1);
2. how to model the reformulation signal (Section 3.2);
3. how to detect drifts in the reformulation signal in an unsupervised manner (Section 3.3).

3.1 Creating User Behavioral Logs

In this Section, we describe how to derive *user behavioral logs* (in Figure 3) from search interaction logs.

We consider the following scenario: we have a stream of queries submitted to a search engine. In response to each query, the engine returns $SERP$. Users may decide to click on one or more $URLs$ in $SERP$, reformulate their queries,

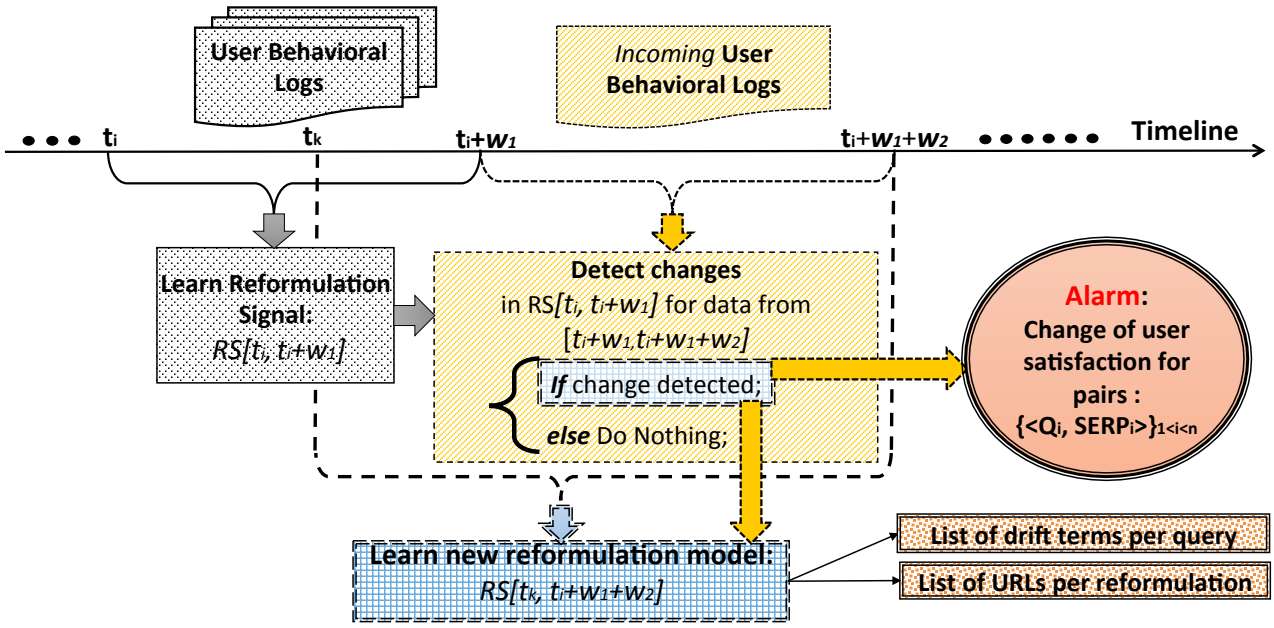

Figure 3: Overview of a framework for detection changes in user satisfaction with search results.

or end their sessions. These types of user interactions are stored in search interaction logs.

We convert standard search interaction logs to the user behavioral logs where we store information only about reformulations of issued queries. We use a query expansion definition from [20], [15] to detect terms which users used for reformulations.

An example of user behavioral entries is presented in Figure 4. Each entry consists of four columns:

- **Session ID** is a session identification information;
- **Timestamp** is a time when an action is performed;
- **Action** is an action type, that a user performed: we record the following action types: search, reformulation, and click on a $SERP$ page .
- **Action details** is details of a user's action: for the search action we record an issued query, for the click action we record an identifier of clicked page, for the reformulation action we record a reformulation term.

For example **'2014'** is a *reformation term* for the initial query *'cikm conference'* if users are looking for up-to-date information about the conference.

User behavioral logs are suitable to collect a dictionary of the reformulation terms:

$$D_Q = \{K_j\}_{j=1}^{n}, \tag{2}$$

where K_j is j^{th} the reformulation term used to change the query Q, n is the number of the reformulation terms used for expanding the query Q.

In the next Section, the dictionary of the reformulation terms will be utilized for modelling the reformulation signal.

Session Id	Timestamp	Action	Action details
.....			
123457	1388494920	search	Query ='flawless'
123457	1388494980	click	Page Id = '755'
123457	1388495060	reformulation	Query ='flawless beyonce' => Reformulation = 'beyonce'
123457	1388495115	click	Page Id = '170'
123458	1388495415	search	Query ='cikm conference'
123456	1388361661	reformulation	Query ='cikm conference' => Reformulation = '2014'
123456	1388361720	click	Page Id = "45"
.....			

Figure 4: An example of user behavioral log.

3.2 Modelling The Reformulation Signal

In this Section, we describe how to build a reformulation signal model, that is presented in our framework in Figure 3 as *'Learn Reformulation Signal'*.

We build the reformulation signal (RS) of queries for a time period $[t_i, t_i + w_1]$, using the user behavioral logs. RS of the query Q would be:

$$RS = \left\{P_{[t_i, t_i+w_1]}\left(K_j, Q\right)\right\}_{j=1}^{n}, \tag{3}$$

where w_1 is the selected size of the inference window, $P\left(K_j, Q\right)$ is a joint distribution of the query Q and its reformulation term K_j during the time period $[t_i, t_i + w_1]$.

When time $(t_i + w_1 + w_2)$ comes we rebuild the reformation signal of Q for the time period $[t_i + w_1, t_i + w_1 + w_2]$ using Equation 3:

$$RS = \left\{P_{[t_i+w_1, t_i+w_1+w_2]}\left(K_j, Q\right)\right\}_{j=1}^{m}, \tag{4}$$

where w_2 is the size of a test window.

The presented model for the reformulation signal will be used to detect changes in user satisfaction in the next Section.

3.3 Detecting a Drift in Reformulation Signal

In this section we present an algorithm for detecting a drift in user satisfaction using the reformulation signal. Our goal is to detect statistically significant changes. This action is depicted in Figure 3 as *'Detect Changes'*.

Let us introduce a definition of *a drift in the reformulation signal* between two periods at time $[t_i, t_i + w_1]$ and $[t_i + w_1, t_i + w_1 + w_2]$ using Equations 3 and 4:

$$\exists Q' : P_{[t_i,t_i+w_1]}(K_j, Q) \neq P_{[t_i+w_1,t_i+w_1+w_2]}(K_j, Q), \quad (5)$$

where $P_{[t_i,t_i+w_1]}(K_j, Q)$ denotes the joint distribution of query Q and its reformulation term K_j at the time period $[t_i, t_i + w_1]$.

It is important to determine what it means when the distribution has changed. If the drift in the reformulation signal is statistical significant then *we assume that user satisfaction with the following pair has decreased dramatically*:

$$\langle Q_{[t_i+w_1,t_i+w_1+w_2]}, SERP_{[t_i,t_i+w_1]} \rangle, \quad (6)$$

where $Q_{[t_i+w_1,t_i+w_1+w_2]}$ is the query issued at the time period $[t_i + w_1, t_i + w_1 + w_2]$; $SERP_{[t_i,t_i+w_1]}$ is search results, that were generated for Q at the time period $[t_i, t_i + w_1]$, and it is still shown during the time period $[t_i + w_1, t_i + w_1 + w_2]$.

However, users are no longer satisfied and they reformulate Q using some drift term K_j. The fact that the drift has happened at time $(t_i + w_1 + w_2)$ can be a signal that we need to generate a new $SERP_{[t_i+w_1,t_i+w_1+w_2]}$ for Q to improve user satisfaction.

Let us consider an example of a drift in the reformulation signal for the query *'cikm conference'* in Figure 5. Users were satisfied with $SERP$ that was returned by the query *'cikm conference'* at time t_i. However, at time $t_i + \Delta t$ a probability to reformulate the query has been changed dramatically and the term *'2014'* is the most frequent reformulation. Most likely users have changed their behavior due to an upcoming conference event and they could not find the right link in $SERP$ that was optimized for clicks from the last year.

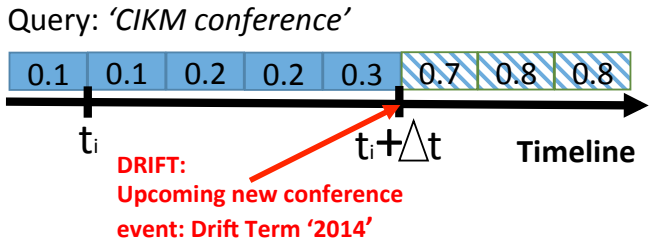

Figure 5: Example of concept drift in probability to reformulate the query *'CIKM conference'* using drift term *'2014'*.

The proposed algorithm $DDSAT$ for detecting changes in the query reformulation signal to discover changes in user satisfaction is presented in Algorithm 1. We will explain how $DDSAT$ works next.

Let us clarify which an input and an output $DDSAT$ has:

- ***DDSAT* Input.** Our current implementation of the algorithm is using a fixed size of the inference window w_1 that equals to one month. We also experiment with two sizes of the test window w_2: two weeks and one week. We calculate an error threshold e, using the following formula as described in [5]:

$$e = \sqrt{\frac{1}{2m} * \sigma_W^2 * \ln\frac{4}{\delta'}} + \frac{2}{3m}\ln\frac{2}{\delta'}, \quad (7)$$

  where m is a harmonic mean of $||w_1||$ and $||w_2||$, σ_W^2 is the observed variance of the elements in window $W = w_1 \cup w_2$ and $\delta' = \frac{\delta}{n}$, δ is a confidence value and n is a total size of two windows. For experimentation, we run our algorithm with the three different confidence values: $\delta = \{0.05, 0.1, 0.3\}$.

- ***DDSAT* Output.** $DDSAT$ returns an alarm as an output if the drift happens.

Let us consider the method $processDetecedConceptDrift()$ in Algorithm 1 that deals with the detected drifts. Moreover, the function $processDetecedConceptDrift()$ has two additional input parameters which show how the observed drifts influence the current system:

- **Parameter 'extra'** is a boolean variable, if it is *'true'* $DDSAT$ will produce two extra statistics:
  1. a list of drift terms, which can be used for serving a fresh query suggestion;
  2. a list of drift $URLs$ which can be used for reranking of $SERP$.
- **Parameter 'update'** is a boolean variable, if it is *'true'* $DDSAT$ will update the reformulation signal.

Algorithm 1 Algorithm for detection drift in user satisfaction using reformulation signal ($DDSAT$).

Require: inference window w_1;
test window w_2;
error threshold e;
start time t_i;
produce extra information $extra \leftarrow true, false$;
learn new reformulation signal $update \leftarrow true, false$;
User Behavioural Log (UBL);
Ensure: $drift \leftarrow true, false$
1: $\{RS_{w_1}(Q_k)\}_{m=1}^{k} \leftarrow$buildRefSignal$(UBL_{[t_i,t_i+w_1]})$
2: $\{RS_{w_2}(Q_k)\}_{m=1}^{k} \leftarrow$buildSignal$(UBL_{[t_i+w_1,t_i+w_1+w_2]})$
3: **for** $Q_m, K_j \in ULB$ **do**
4: **if** $|\mu(P_{w_1}(K_j, Q_m)) - \mu(P_{w_2}(K_j, Q_m))| > e$ **then**
5: $drift \leftarrow true$
6: $processDetecedConceptDrift(extra, update)$
7: **else**
8: $drift \leftarrow false$
9: **end if**
10: **end for**
11: **return** $drift$

The presented algorithm $DDSAT$ works in an unsupervised way, it does not require any human labelling. It can be shown that $DDSAT$ has a linear complexity.

The proposed framework can be used as a monitoring tool, which alarms when user satisfaction changes for a particular pair: $\langle Q, SERP \rangle$. Our framework also gives an explanation for detected changes, it returns the list of drift terms. Moreover, it suggests a possible solution for serving up-to-date $SERP$ and returns the list of the most frequently clicked $URLs$ after reformulating Q using the drift terms.

4. EXPERIMENTAL SETUP

The ultimate goal of the presented framework is to detect changes in user satisfaction. In order to make the evaluation realistic we use search interaction logs of a commercial search engine, that we describe in details next. We run our framework over this dataset. In *DDSAT* algorithm we set to *'true'* the parameters: *'extra'* and *'update'* of the method *processDetectedConceptDrift()*. The extra statistics are used to setup a human assessment. Our evaluation scenario is described in Section 4.2, The final results is presented in Section 5.

4.1 Data

Our experimental data comprises of search interaction logs of a commercial search engine that were collected during six months: *September* 2013*, October* 2013*, November* 2013*, December* 2013*, January* 2014*, February* 2014. We only include log entries for US-based traffic. We derive user behavioral logs from the selected search interaction logs as presented in Section 3.1. Each month of data consists of over 10 million records.

4.2 Evaluation Methodology

In this Section we describe how we organize evaluation of our framework results:

1. the derived list of detected changes with drift terms;
2. the derived list of the most clicked $URLs$ per drift.

First, let us present a format of the presented system output that needs to be evaluated. Our framework returns results in the form presented in Figure 6. It contains:

- **Date -** is a time period when drift happened;
- **Initial Query -** is a query that users issued;
- **Drift term -** is a term that cause the drift because users added it to expand the initial query at time period depicted as *Date*.
- **URL -** is a link that users clicked the most after issuing the reformulation.

4.2.1 Human Drift Judgments

For the evaluation, a group of annotators were instructed to exhaustively examine the detected drift terms. The judges were well trained to understand time-related and event-related drifts in user behavioral data. They were given relevant examples of drift, e.g:

1. The latest Olympic games were held in February 2014 and users run related queries such as *'medals olympics'* (users were looking for information about medals among USA 2014 Olympic team). However, if users were served with results from Olympics 2012 then they had to reformulate the query using the reformulation term *'2014'* in order to find desired results. That is an example of a gradual drift, because users got interested in 2014 Olympics gradually over two years.

2. Another example would be a breaking news query such as *'novak djokovic'* (a Serbian professional tennis player) who got engaged at the end of September 2013. Users were interested in this news and tended to reformulate the query by adding the reformulation term *'fiancee'*. That is an example of a sudden drift. This kind of drifts have quite a short lifetime because users remain interested during a limited period of time. However, it is important to serve it right in time.

Date	Initial query	Drift Term	URL
.....			
Oct. 2013	novak djokovic	fiancee	URL48
Oct. 2013	CIKM conference	2014	URL44
Jan. 2014	flawless	beyonce	URL578
Jan. 2014	feliz ano nuevo	2014	URL48
Jan. 2014	ct 40ez	2013	URL109
Feb. 2014	when is fastnacht day	2014	URL48
Feb. 2014	mormons olympics	2014	URL409
.....			

Figure 6: Example of an output of the framework. Column *URL* is anonymized.

An example of an evaluation task for the annotators is presented in Figure 7 (A). The judges were asked to decide: Can the term T be a drift term for the query Q? The judges were allowed to use external information sources to find answers.

Every discovered drift, characterised by a drift term, is judged by three different annotators using binary classes: *'0' = wrong* and *'1'= right*. The final score is calculated based on three judgements.

We use accuracy as a final evaluation metric that we refer as *Drift Accuracy*.

4.2.2 Human Judgments for drift $URLs$

We calculate a statistic for $URLs$ which users clicked in $SERP'$ that is derived after reformulation Q to Q'. If the probability of clicking on URL is greater than 0.5, then it is drift URL. If clicks are diverse we cannot produce any URL.

Judges, whom we asked to evaluate drift URL, answered the following question: *'Is* URL relevant *for the initial query* Q *at the time period* T*?'* In other words, we asked human annotators to evaluate a tuple $\langle Q, URL, T \rangle$ as proposed in the paper [7]

Every discovered drift URL is also judged by three different persons using binary classes. The final score is calculated based on three judgements. We use accuracy as a final evaluation metric for drift URL, that we refer as *URL Accuracy*.

We described the large-scale evaluation of our method based on real dataset from a commercial search engine that was collected during six months. As we will show next we

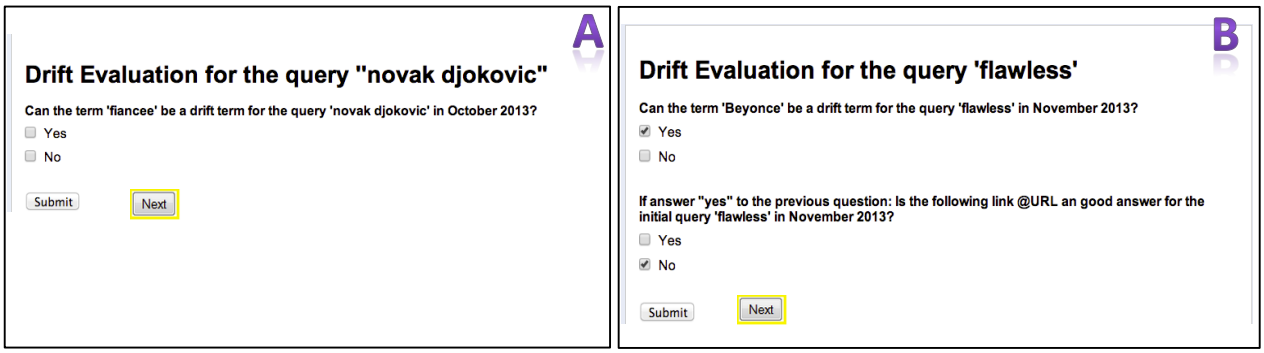

Figure 7: Two of fragments of an evaluation task for the annotators: (A) is the task when we do not have most clicked *URL* because click are diverse and (B) is the task when we can suggest *URL* and (B).

can precisely identify drift in user satisfaction using the reformulation signal.

5. EXPERIMENTAL RESULTS

We now summarize the results for detection changes in user satisfaction using the reformulation signal. The proposed solution is working in an unsupervised way and can be applied to large search interaction logs. As a ground truth, we use the human judgements that are described in Section 4.2.

5.1 Defining Sizes of Inference and Test Windows

It is important to note that we fix a size of the inference window w_1 to one month of data. For the test window w_2, we experiment with three different sizes: one week, two weeks, one month. As a final result, we will report for w_2 equals two weeks.

For our experimentation we use data described in Section 4.1, that can be easily transformed to user behavioral logs. The algorithm *DDSAT*, proposed in Algorithm 1, is running on derived data in the following way:

1. *DDSAT* starts at time t_i (for our datasets: t_i equals to 1^{st} *of September*);
2. *DDSAT* builds the reformulation signal (*RS*) based the time period $[t_i, t_i + w_1]$ (for our datasets: the reformulation signal is built on September 2013);
3. *DDSAT* detects drifts in *RS* based on the time period $[t_i + w_1, t_i + w_1 + w_2]$ and produces a list of detected drifts (for our datasets: first two weeks of October 2013);
4. *DDSAT* reassigns t_i to $(t_i + w_1)$ and goes to *1.*

We combine all detected drifts and drift *URLs* and evaluate them using methodology described in Section 4.2.

However, for other domain the size of w_1 and w_2 are dependent on many aspects such as a volume of a traffic, type of a served content and so on. Implementers with a domain's knowledge should decide how often run our framework. However, we plan to extend the algorithm *DDSAT* so that it can determine when there was a drift on the fly.

5.2 Defining confidence value

We evaluate the discovered list of the drifts in user satisfaction to check which confidence level suits best our needs. The randomly selected part (30%) of human judgements for detected drifts (Section 4.2.1) are used to calculate the accuracy below. Our findings are the following:

- for the confidence value $\delta = 0.05$ accuracy is 65%;
- for the confidence value $\delta = 0.1$ accuracy is 68%;
- for the confidence value $\delta = 0.3$ accuracy is 66%;

The rest of judgements (70%) are used to calculate a final accuracy for our drift detection method in Section 5.3.1.

Hence, we use the confidence value $\delta = 0.1$ for the future evaluation because it gives us the highest accuracy.

5.3 Evaluating DDSAT

In this section we describe the experiments we conducted to evaluate the accuracy of our method *DDSAT*. We evaluate two types of the accuracy:

1. in Section 5.3.1, we present the accuracy of overall drift detection that is calculated based on 70% of human judgements collected in Section 4.2.1;
2. in Section 5.3.2 we demonstrate the accuracy of detected drift *URLs* that is calculated based on annotator's judgements collected in Section 4.2.2.

5.3.1 Drift Accuracy

Drift accuracy is a percentage of times when the drift in user satisfaction is correctly detected using the reformulation signal. We calculate drift accuracy with respect to the number of users who issue the reformulation. The obtained accuracy is presented in Table 1.

Of course, the best result is characterized by the greatest amount of users. Rows in Table 1 for the number of users in a range $[800, 1000)$ and $[500, 800)$ have the lower accuracy than the accuracy rate for the number of users in the range $[250, 500)$ because they include smaller number of detected drifts in user satisfaction.

Table 1: The accuracy of the drift detection depends on the number of users who issued reformulations. The metrics are calculated based on the results obtained with the confidence value $\delta = 0.1$.

Number of Users	Drift Accuracy
$[1000, 1300)$	98%
$[800, 1000)$	67%
$[500, 800)$	80%
$[250, 500)$	82%
$[1, 100)$	66%

Table 2: The accuracy of drift URL depending on the number of users who issued reformulations. The metrics are calculated based on results obtained with confidence value $\delta = 0.1$

Number of Users	Drift URL Accuracy
$[1000, 1300)$	100%
$[800, 1000$	81%
$[500, 800)$	85%
$[250, 500)$	91%
$[1, 100)$	87%

5.3.2 *Drift URL Accuracy*

Drift URL accuracy is a percentage of relevant $URLs$ among the list of proposed drift $URLs$. The obtained accuracy is presented in Table 2. The quality of derived drift $URLs$ is very high especially for the number of users greater than 250. We see in Table 2 the same situation as in Table 1 for the number of users in a range $[800, 1000)$ and $[500, 800)$. They have lower accuracy than the number of users in the range $[250, 500)$ because they include a smaller number of detected drift $URLs$.

Our framework includes URL into the list of drift $URLs$, if the probability of clicking on them after reformulating is higher than 0.5. Potentially, detected drift $URLs$ can be applied directly into a learned ranking function as a *'freshness feature'* or used for a re-ranking of current results.

To summarize our evaluation of $DDSAT$, we recommended to determine a confidence value for the drift detection that gives highest accuracy of the detected drift. In our case, we obtained the confidence value $\delta = 0.1$. The propose algorithm was evaluated from two points of view:

1. the accuracy of the detected drift in user satisfaction is high and it gets especially precise if the number of users who issued reformulation is greater that 250 (Table 1). We do not report the row '> 1000' because it not always realistic for smaller search engines;

2. the accuracy of how relevant are detected drift $URLs$. It is especially accurate if the number of users who issued reformulation is greater that 250 (Table 2).

5.4 Detecting Anomalies in Results

In this Section we show an example of *outliers* in a concept drift and how to deal with this kind of anomalies.

User behavior the Web is not always reliable, it can be sometimes spurious. It is important for the algorithm $DDSAT$ to know how to remove this anomalies from user behavioral logs data in order to return more accurate results of drift detection.

For instance, *'spurious behavior'* can be caused by Search Engine Optimization (SEO) that is a process of affecting the visibility of websites or web pages in search results of search engines. In general, SEO aims to push a site to a higher rank on the search results page, and more frequently a site appears in the search results list, the more visitors it will receive from the search engine's users. In order to achieve the goal, SEO considers how search engines work, what people search for, the actual search terms or keywords typed into search engines and which search engines are preferred by their targeted audience. Optimizing of a website may involve editing its content, HTML and associated coding to both increase its relevance to specific keywords and to remove barriers for indexing activities of search engines.

While we were analysing the list of derived drifts we noticed abnormal drifts, e.g. the query *'aol mailbox sign in'* was reformulated using the drift term *'agnes corky'*. The reformulation was issued by more than 200 users. However, this drift did not make any sense. This behavior most probably was simulated. However, we noticed that only one click happened and that clicked page referred to the website with the domain named *'seotest'*. Hence, we concluded that it is the anomaly.

For the final results, this kind of anomalies need to be filtered out. They are removed by using the following heuristic rule: *'If the number of users who issued reformulation:* (U) *is much greater than the number of user clicks on SERP' (search results after reformulating an initial query):* $(Click_U)$ *then the detected change is the Anomaly':*

$$\text{if } U \gg Click_U \text{ then } Anomaly. \tag{8}$$

To summarise our experimental results, the proposed technique for detecting changes in user satisfaction using the reformation signal works well on real datasets. The observed results over large datasets (all traffic from a commercial search engine during 6 months) are both substantial and statistically significant. Furthermore, we have shown that results of our framework, such as lists of $URLs$, can be potentially useful for ranking.

6. APPLICATIONS

In this Section we discuss potential applications where the results of the developed framework can be used.

6.1 Learning to Rank

A key component of our system is the algorithm $DDSAT$ that is monitoring user engagement. It alarms when changes in user satisfaction happen with the pair $\langle Q, SERP \rangle$. $DDSAT$ alarm is a signal that user intent for Q drifted and we need to change $SERP$ to satisfy changes in user needs. Potentially, the detected drift can be applied directly into a learned ranking function as a *'freshness feature'* or used for re-ranking. Moreover, our framework produces the list of $URLs$, which users prefer after reformulating the initial query. This list also can be incorporated into a ranking model.

6.2 Query Auto-Completion

Query auto-completion is an important feature of online search engines that enhances search experience by saving users time which otherwise would be spent on typing. A time-sensitive approach has been proposed in [29] for query auto-completion. Our framework also returns drift terms,

which are reformulation terms that cause a drift in the reformulation signal. Hence, this list can be used for time-sensitive query auto-completion.

6.3 Automatically Detecting Underperforming Queries

Automatic detection of problematic queries, where search engines do not return a required result and users are dissatisfied with their search results, has been extensively studied [1], [9], [12], [23]. However, previous work largely utilise user interaction features, topical and lexical attributes to detect such underperforming queries. Time-sensitive nature of user satisfaction has not been considered.

In this paper, we propose the method to identify drifts in user satisfaction over time. The proposed framework monitors a system and it signals an alarm when drift in user satisfaction with the pair $\langle Q, SERP \rangle$ happens. Hence, when we know a problematic query we can retrain our ranker in order to improve quality of retrieved $SERP$. We can use the engagement on the reformulated query in order to derive training pairs.

7. CONCLUSION AND FUTURE WORK

In this Section we conclude and present our view of future work.

7.1 Conclusion

In this paper we explored the utility of incorporating the query reformulation signal in detecting changes in user satisfaction. We leverage concept drift techniques to detect changes in user satisfaction with the pair $\langle Q, SERP \rangle$ over time due to some events. The appearance of a drift requires a modification of the $SERP$ to satisfy shifted user needs. We introduced a novel *Drift Detection in user SATisfaction (DDSAT)* algorithm, that accurately detects changes. The proposed algorithm works in an unsupervised manner, it does not need any labelled data. *DDSAT* is a part of the developed framework for detecting changes in user satisfaction.

We conducted a large-scale evaluation using data from a commercial search engine. The dataset was collected during six months. Our experiments show that the algorithm *DDSAT* works with a high accuracy. Moreover, our framework outputs the list of drift terms and the list of $URLs$, which can be used for the future re-ranking of $SERP$.

The algorithm of the drift detection in user satisfaction which we presented in this paper can be incorporated in many search-related applications where freshness is required, e.g. in recency ranking, query auto-completion.

7.2 Future Work

We believe that the current implementation of the algorithm *DDSAT* can be improved. The algorithm uses the fixed sizes of the inference and test windows. However, it is not always suitable. For instance, the size of the test window for the sudden drift can be way shorted compared to incremental drifts. We anticipate that the size of the test window should be proportional to the reformulation frequency. We would like to develop a method to identify dynamically the size of the inference and test windows.

We also would like to identify the type of the detected drift in Figure 2. It is important to know in order to define the lifetime. If our algorithm detects the sudden drift (e.g. breaking news queries) then its lifetime is much shorter compared to incremental or sudden drifts. We would like to develop a method to identify automatically the type of the drift.

ACKNOWLEDGMENTS

We thank Nick Craswell, Andreas Bode, Jonas Barklund, Paul De Bra and Mykola Pechenizkiy for fruitful discussions and their valuable suggestions, and Willi Richert for providing help with infrastructure for experimentation. We also thank Ahmed Hassan for providing the code for reformulation detection.

This research has been partly supported by STW and it is the part of the CAPA[1] project.

References

[1] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein. Find it if you can: a game for modeling different types of web search success using interaction data. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, 2011.

[2] E. Agichtein, E. Brill, and S. T. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 19–26, 2006.

[3] E. Agichtein, E. Brill, S. T. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 3–10, 2006.

[4] A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 773–774, 2007.

[5] A. Bifet and R. Gavaldá. Learning from time-changing data with adaptive windowing. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2007.

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

[7] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, and C. L. F. Diaz. Towards recency ranking in web search. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 11–20, 2010.

[8] A. Dries and U. Ruckert. Adaptive concept drift detection. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, pages 233–244, 2009.

[9] H. A. Feild, J. Allan, and R. Jones. Predicting searcher frustration. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 34–41, 2010.

[1] www.win.tue.nl/∼mpechen/projects/capa/

[10] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 2013.

[11] Q. Guo, R. W. White, Y. Zhang, B. Anderson, and S. T. Dumais. Why searchers switch: understanding and predicting engine switching rationales. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 335–344, 2011.

[12] A. Hassan, R. Jones, and K. L. Klinkner. Beyond dcg: user behavior as a predictor of a successful search. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 221–230, 2010.

[13] A. Hassan, X. Shi, N. Craswell, and B. Ramsey. Beyond clicks: query reformulation as a predictor of search satisfaction. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2019–2028, 2013.

[14] S. Hido, T. Idé, H. Kashima, H. Kubo, and H. Matsuzawa. Unsupervised change analysis using supervised learning. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 148–159, 2008.

[15] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 77–86, 2009.

[16] S. B. Huffman and M. Hochster. How well does result relevance predict session satisfaction? In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 567–574, 2007.

[17] D. B. in Web Search Samuel Ieong, D. B. in Web Search Samuel Ieong, N. Mishra, E. Sadikov, and L. Zhang. Domain bias in web search. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 55–64, 2012.

[18] Y. Inagaki, N. Sadagopan, G. Dupret, A. Dong, C. Liao, Y. Chang, and Z. Zheng. Session based click features for recency ranking. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2010.

[19] Y. Inagaki, N. Sadagopan, G. Dupret, C. L. A. Dong, Y. Chang, and Z. Zheng. Session based click features for recency ranking. In *Association for the Advancement of Artificial Intelligence*, 2010.

[20] B. J. Jansen, D. L. Booth, and A. Spink. Patterns of query reformulation during web searching. *JASIST*, 60(7):1358–1371, 2009.

[21] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[22] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 154–161, 2005.

[23] Y. Kim, A. Hassan, R. W. White, and Y.-M. Wang. Playing by the rules: mining query associations to predict search performance. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 133–142, 2013.

[24] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, pages 167–176, 2011.

[25] K. Radinsky, K. Svore, S. T. Dumais, J. Teevan, A. Bocharov, and E. Horvitz. Modeling and predicting behavioral dynamics on the web. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 599–608, 2012.

[26] T. Saracevic. Relevance: A review a the literature and a framework for thinking on the notion in information science. part iii: Behavior and effects of relevance. *JASIST (JASIS)*, 58(13):2126–2144, 2007.

[27] T. Saracevic. Relevance: A review of the literature and a framework for thinking on the notion in information science. part ii: nature and manifestations of relevance. *JASIST (JASIS)*, 58(13):1915–1933, 2007.

[28] J. C. Schlimmer and R. H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of AAAI Conference on Artificial Intelligence*, 1986.

[29] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proceedings of the ACM Conference on Research and Development on Information Retrieval (SIGIR)*, pages 601–610, 2012.

[30] R. W. White and S. T. Dumais. Characterizing and predicting search engine switching behavior. In *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, pages 87–96, 2009.

[31] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning (ML)*, 23(1):69–101, 1996.

[32] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of International World Wide Web Conferences (WWW)*, pages 1011–1018 1011–1018, 2010.

[33] I. Zliobaite. Learning under concept drift: an overview. *CoRR abs/1010.4784*, 2010.