

---

# Using Clickthrough Data to Improve Web Search Rankings

---

Christoph Zimmerli, [zimmerch@ethz.ch](mailto:zimmerch@ethz.ch)

Algorithms for Data Base Systems Seminar,  
SS07, ETH Zürich

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Relevance Feedback</b>	<b>2</b>
2.1	Explicit Relevance Feedback . . . . .	2
2.2	Implicit Relevance Feedback . . . . .	2
2.3	Clickthrough Data . . . . .	2
<b>3</b>	<b>User Behaviour</b>	<b>3</b>
3.1	Eye Tracking . . . . .	3
3.2	Trust Bias . . . . .	4
3.3	Quality of Context Bias . . . . .	4
<b>4</b>	<b>Extracting Preference Statements</b>	<b>4</b>
4.1	Strategies . . . . .	5
<b>5</b>	<b>Query Chains</b>	<b>6</b>
5.1	Strategies . . . . .	7
<b>6</b>	<b>The Learning Algorithm</b>	<b>8</b>
6.1	A Basis for Learning an Improved Ranking . . . . .	8
6.2	The Ranking SVM Algorithm . . . . .	8
6.3	The mapping $\Phi$ . . . . .	9
6.3.1	Contents of $\Phi$ (Version 1) . . . . .	9
6.3.2	Tests & Results . . . . .	10
6.3.3	Analysis of the Learned Function . . . . .	11
6.3.4	Contents of $\Phi$ (Version 2) . . . . .	11
6.3.5	Tests & Results . . . . .	12
<b>7</b>	<b>Evaluating the Robustness of the Learning Method</b>	<b>12</b>
7.1	Document Generation . . . . .	12
7.2	User Model . . . . .	12
7.3	Experiments . . . . .	13
7.3.1	Accuracy of Relevance Estimates . . . . .	13
7.3.2	Topic and Word Ambiguity . . . . .	13
7.3.3	User Trust in the Ranking . . . . .	13
7.3.4	Number of Results Looked at . . . . .	13
7.3.5	Query Reformulations . . . . .	13
7.4	Conclusions . . . . .	14
<b>8</b>	<b>Collecting Unbiased Preferences from Clickthrough Data</b>	<b>14</b>
8.1	Presentation Bias . . . . .	14
8.2	FairPairs . . . . .	14
8.3	Conclusions . . . . .	14
<b>9</b>	<b>Wrap Up</b>	<b>15</b>
<b>10</b>	<b>References</b>	<b>15</b>

## 1 Introduction

The results of the annual Text Retrieval Conference (TREC) [1] have shown that there is no one-fits-all ranking function for text retrieval.

This implies the need for adaptation to specific (groups of) users and/or document collections to get the best results for each setting. This can either be done by building different retrieval functions for different settings, or by starting with a general function and adapting it to the needs of the users via their feedback.

The work presented in this report is all based on the latter idea.

To succeed in such an adaptation process, the following key ingredients are necessary:

- Relevance feedback on the presented ranking
- Extraction of preference constraints from the feedback
- Adaptation of the ranking function
- Testing the new function

Section 2 explores the possibilities for collecting relevance feedback. Then we will take a look at user behaviour in section 3 to understand the feedback. We are then able to come up with strategies for extracting preference constraints in the sections 4 and 5. To learn from these constraints section 6 presents a Support Vector Machine (SVM) algorithm and evaluates its performance. Section 7 then shows an approach to test such a learning algorithm for different document collections and user behaviour. Finally, section 8 presents a method to collect better feedback from the users, before section 9 wraps up of all the topics covered in this report.

## 2 Relevance Feedback

This section discusses two basic forms of relevance feedback: explicit and implicit relevance feedback. After that, the process of recording clickthrough data will be presented.

### 2.1 Explicit Relevance Feedback

Explicit relevance feedback requires a human to explicitly state which documents are relevant to a given query and which are not. While explicit feedback provides clean data, it requires the user to do additional work when using the search engine. Not all users are ready to do this. If special relevance judges are employed to do the additional work, the process is more time consuming and more costly.

### 2.2 Implicit Relevance Feedback

Collecting implicit relevance feedback is done by observing a user's behaviour while he interacts with the search engine. There is no additional work for humans. The behavioural data can be gained from log files, which is a vast resource in the world of web search engines that is available for free, and can be processed by computers.

### 2.3 Clickthrough Data

Clickthrough data refers to the links that a user clicks on when presented the result of a web search. This data can be recorded in log files, for instance by using a transparent proxy server which records the clicks, or by asynchronously sending the information about a click to the search engine. These clicking decisions are saved together with the issued query and the

presented result. To gain knowledge from the recorded data it is necessary to understand the user's behaviour.

### 3 User Behaviour

The key to improving a ranking function from clickthrough data is to understand the user who produced the data. That's why a study with students from Cornell University was conducted in [2].

They were each given 10 questions that had to be answered using Google as search engine. There were two phases in the study. In phase 1 the normal Google ranking was returned to the users. In phase 2 users were shown one of 3 possible rankings:

- normal: the ranking that Google returned
- swapped: same as normal, except that the top 2 links were swapped in position
- reversed: the reverse order of the Google ranking

The users were not aware of the manipulation of rankings.

The number of queries per question and clicks per query were about the same for both phases.

To find out how good the generated implicit feedback was, explicit relevance judgements were collected from another group of people. They had to order all links contained in a result from Google according to their perceived relevance based on the abstract, without knowing Google's ranking. It is known that humans are better at ordering things in such a way, than giving absolute relevance values. For phase 2 judgements based on the actual websites behind the links in the results page were collected as well.

The evaluation of these explicit judgements showed a good inter-judge agreement, so they can be used as benchmark for the generated implicit feedback.

#### 3.1 Eye Tracking

The study employed eye tracking to find out what parts of a results page of Google users actually look at before clicking on a link. Evaluating the eye tracking data, only the so-called fixations were analysed. Fixations are longer time spans where the eyes look at the same region. This is widely accepted to be the behaviour where most information acquisition and processing occurs.

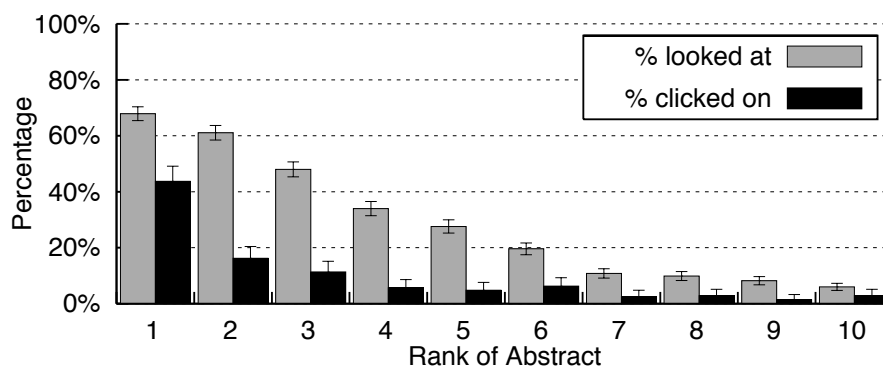


Figure 1: Percentage of time an abstract was viewed or clicked depending on the rank of the abstract. Figure taken from [3].

Figure 1 shows that users are most likely to look at the top 2 results. They see those almost the same amount of times. Nevertheless does the first result draw much more clicks than the second. This interesting behaviour will be discussed later. Also visible in the figure is a drop in attention at scroll border that occurs around rank 6 and 7. It seems that only a part of all users even does scroll down. But once they do, they look at all the remaining links on the page, which results in the almost equally distributed percentage of fixations beyond rank 7.

An evaluation of the arrival time of a fixation at a certain abstract shows that on average users scan the results from top to bottom.

It was also found that, as reported before by other researchers, users follow a depth-first search strategy: they click on a promising link without first looking at the rest of the results.

Two other patterns identified are that abstracts close above the one clicked are more likely to be viewed, than others further above, and that the abstract right below the clicked one is also likely to be seen before the click.

Often a click on a link is viewed as an absolute statement: this page is relevant. However, there are at least two factors indicating that this is not absolutely true: Trust bias and quality of context bias. These two phenomena will be discussed in the following.

### 3.2 Trust Bias

The results ranked 1 and 2 are looked at about equally often, but the top result is favoured to click on. Is this because Google's ranking is that good and the top link is mostly the better match to the query than the second? Or do users prefer the first link due to trust in the ranking ability of the search engine?

Looking at the results from the "swapped" setup in phase 2 it becomes obvious that users click on the first link, even if it is less relevant according to Google's original ranking. This means that the position of a result actually influences a user's decision to click on it. Therefore users trust the search engine to correctly order the documents by relevance to a query.

### 3.3 Quality of Context Bias

Comparing the results from the "normal" and the "reversed" settings, the users clicked on average on less relevant documents in the latter case. This shows that users try to find the most relevant document in the presented ranking, even if it is no total match.

Another observation is that the average rank of a clicked document goes up from 2.66 in the "normal" setting to 4.03 in "reversed". This means that users look further down on the result page to find the more relevant documents.

Combining these two facts, it can be said that the user's selection is influenced by the overall quality of the presented ranking, and a click is only a relative relevance measure for the results the user has actually seen.

As the decision to click is based on the abstract presented with a link, clicks are also influenced by the quality of these abstracts. If a document is relevant but this is not visible because the abstract was poorly chosen, the user will skip it and look for a better result. Turning it around one can say that user behaviour can be used to measure the quality of a ranking and the relevance of individual documents to a query.

## 4 Extracting Preference Statements

Now that the user's interactions with the result page are known, we can come up with strategies to generate preference statements from clickthrough data.

For instance if a user is presented with a ranking of six links  $l_i$  in descending order

$$l_1 \quad l_2 \quad l_3 \quad l_4 \quad l_5 \quad l_6$$

and he clicks on the links

$$l_1 \quad l_3 \quad l_5$$

we can derive, that  $l_3$  is more relevant than  $l_2$  as the user has probably seen  $l_2$  and then intentionally skipped it.

Naming the relevance of a link  $l_i$  with  $rel(l_i)$  one can derive the following pairs of relevance information:

$$rel(l_3) > rel(l_2), \quad rel(l_5) > rel(l_2), \quad rel(l_5) > rel(l_4)$$

Using such partial relevance judgements it is possible to derive several interpretation strategies that will be presented in the following.

To measure the accuracy of these strategies, the agreement of the pairwise preferences with the explicit feedback from the judges is used. The results are shown in table 1. The ‘‘Inter-Judge Agreement’’ row can be seen as a benchmark of what accuracy is possible at best.

Explicit Feedback for Data Strategy	Abstracts					Pages
	Phase 1 ‘‘normal’’	Phase 2			all	Phase 2 all
	‘‘normal’’	‘‘swapped’’	‘‘reversed’’			
Inter-Judge Agreement	89.5	N/A	N/A	N/A	82.5	86.4
Click > Skip Above	80.8	88.0	79.6	83.0	83.1	78.2
Last Click > Skip Above	83.1	89.7	77.9	84.6	83.8	80.9
Click > Earlier Click	67.2	75.0	36.8	28.6	46.9	64.3
Click > Skip Previous	82.3	88.9	80.0	79.5	81.6	80.7
Click > No Click Next	84.1	75.6	66.7	70.0	70.4	67.4

Table 1: Accuracy of several strategies for generating pairwise preferences from the result of one query. Base of comparison are either the explicit judgments of the abstracts or the page itself.

## 4.1 Strategies

### Click > Skip Above

This is the strategy from the example above.

With 80% agreement with the human judges in phase 1, and 83% for the abstract and 78% for the pages in phase 2, the accuracy is quite good.

### Last Click > Skip Above

The intuition is that later clicks on a result are based on more information than the earlier ones. The statistics show this to be true, which results in a slightly better ranking than from ‘‘Click > Skip Above’’. Note that the preferences generated here are a subset of those from ‘‘Click > Skip Above’’.

### Click > Earlier Click

Here the assumption is that later clicks are on more relevant abstracts than earlier ones. This is not well supported by the statistics - the accuracy is worse than for ‘‘Click > Skip Above’’. There are also big differences between the ‘‘normal’’ and ‘‘reversed’’ settings visible. The increased amount of scanning in the latter case seems to lead to well informed choices already for the early clicks.

**Click > Skip Previous**

When clicking on a link, the user has very likely also seen the one right above, and then decided to skip it.

This also generates a subset of the preferences of “Click > Skip Above” and reaches about the same level of accuracy as that strategy.

**Click > No Click Next**

Eye tracking has shown that the link right below the clicked one is also often looked at. This generates a preference statement if the one below does not get a click.

With 84% accuracy this strategy is very good in “normal” but only affirms the initial ranking in that condition, which leads to better than random performance even if the user behaves randomly. In “reversed” results are less promising than “Click > Skip Above”. This is the only strategy that produces preferences that are aligned with the initial ranking, all others are opposed to it.

## 5 Query Chains

Studies of web search engine logs show that a user issues between 1.6 and 2.8 queries during a session, depending on the definition of a session. Causes can be the discovery of an unexpected ambiguity of some query terms or an empty result set. This shows that analysing such query chains could generate additional preferences. These preferences would help to present links that the user clicked on after query reformulation already in the result set of the first query. It is not assumed that preferences extracted from query chains are more accurate than the above within-query strategies, but they lead to more information that was not accessible before.

In the study it is known by construction which queries belong to the same chain. Using machine learning based on features like overlap of query words, overlap and similarity of the retrieved results and time between queries the segmentation into chains proved also feasible in an experiment with the intranet search engine of the Cornell University Library.

It remains however open if this approach will scale for a web search engine.

Explicit Feedback for Data Strategy	Abstracts				Pages
	“normal”	“swapped”	“reversed”	all	Phase 2 all
Click > Skip Earlier QC	84.5	71.1	54.6	70.2	68.0
Last Click > Skip Earlier QC	77.3	80.0	42.1	68.7	66.2
Click > Click Earlier QC	61.9	51.2	35.3	50.6	65.8
Click > Top One, No Click Earlier QC	86.4	77.3	92.6	83.9	85.4
Click > Top Two, No Click Earlier QC	88.9	80.0	86.8	84.2	84.5
Top One > Top One Earlier QC	65.3	68.2	75.6	69.4	69.4

Table 2: Accuracy of several strategies for generating pairwise preferences from query chains. Base of comparison are either the explicit judgments of abstracts or the page itself.

New strategies to exploit query chains will be presented in the following. Some of them are adaptations of the previously discussed within-query strategies. The accuracy of the query chain strategies is shown in table 2.

## 5.1 Strategies

### Click > Skip Earlier QC

This is the extension of “Click > Skip Above” to query chains. A link skipped in an earlier query is supposed to be less relevant than one clicked in a later query.

The accuracy is with 84% good in the “normal” setting, but in “reversed” not significantly different from random baseline.

### Last Click > Skip Earlier QC

This is the subset of preferences from “Click > Skip Earlier QC” that is generated from the last click in a query chain. It performs best in “swapped”. The accuracy for “reversed” is again bad with less than 50%.

Both strategies above share the same weakness: when deciding to click in a later query the user no longer sees the results of the previous queries for direct comparison. He therefore has to make a memory-based decision, which is generally believed to be less accurate.

### Click > Click Earlier QC

The idea is to explore the relationship between clicks in different queries. Clicks in later queries are rated more relevant than clicks in earlier queries.

The accuracy of this strategy is not statistically different from pure random.

A problem of “Click > Skip Earlier QC” as well as of “Last Click > Skip Earlier QC” is that there will only be preferences generated if earlier queries did receive a click. However, 40% of all queries in phase 2 did not receive any clicks. The eye tracking study showed, that users probably looked at the top 2 results before deciding to reformulate their queries. This will be exploited in the next two strategies to overcome the mentioned limitation.

### Click > Top One, No Click Earlier QC

Here clicks in later queries are preferred over first ranked links in previous queries that did not receive a click.

### Click > Top Two, No Click Earlier QC

This prefers clicks in later queries over first and second ranked links in previous queries that did not receive clicks.

These two “Click > Top X” strategies prove to be highly accurate, which leads to the assumption that users actually learn how to improve a query from the unsatisfying results of an earlier query.

The following strategy serves as a test for this assumption.

### Top One > Top One Earlier QC

This strategy explores how often the top result of a later query is more relevant than the top result of an earlier query.

Statistics show the above assumption to be true, which means that the relevance of the top result usually improves through query reformulation. However, the preferences generated with this strategy depend strongly on the currently used retrieval function that decides which document to put on the top spot of a ranking. Therefore this strategy is probably not useful for generating training samples to learn a better retrieval function.

The numbers in the pages column of tables 1 and 2 show, that the generated preference statements not only correspond to judgements based on the abstracts. Moreover, they also correspond to the judgements of the relevance of the actual page behind the link.

Still open is the question which of all the presented strategies can or should be combined to get the best results. To answer this, it is necessary to know how “informative” the preferences generated by the different strategies are. For example do all preferences generated by “Click > No Click Next” affirm the current ranking. Therefore the usage of only this strategy would never lead to changes in the ordering of the results.

## 6 The Learning Algorithm

This section first lays a basis for learning an improved ranking. Afterwards, the pairwise preference statements generated from the strategies in section 4 and 5 will be used to learn an improved ranking function.

### 6.1 A Basis for Learning an Improved Ranking

An optimal retrieval function should for a given query  $q$  and a document collection  $D = \{d_1, \dots, d_m\}$  return a ranking  $r^*$  that orders the documents in the collection according to their relevance to the query. For a strict ordering, this means that for all pairs  $(d_i, d_j) \in D \times D$  either  $d_i < d_j$  or  $d_j < d_i$  is satisfied in  $r^*$ .

To measure the similarity of an existing ranking  $r_f$  and  $r^*$ , Kendall’s  $\tau$  can be adapted. This measure compares the number  $P$  of concordant pairs and the number  $Q$  of discordant pairs (inversions) for the two rankings  $r^*$  and  $r_f$ . For a finite domain  $D$  of  $m$  documents and strict orderings, the sum of  $P$  and  $Q$  is:

$$\binom{m}{2}$$

In this case, Kendall’s  $\tau$  can be defined as:

$$\tau(r^*, r_f) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}}$$

The goal of minimising the number of inversions  $Q$  means that  $\tau(r^*, r_f)$  has to be maximised. This again is equivalent to minimising the average rank of the documents relevant to the query  $q$ .

### 6.2 The Ranking SVM Algorithm

To rank a document  $d$  by relevance to a query  $q$  a linear ranking function  $rel(d, q)$  will be used:

$$rel(d, q) = w * \Phi(d, q)$$

It consists of:

- a weight vector  $w$  to be learned by the algorithm
- a mapping  $\Phi$  onto features describing the match between  $q$  and  $d$

The  $*$ -operator denotes the inner product of the vectors  $w$  and  $\Phi$ . How to choose the contents of  $\Phi$  will be discussed later.

As we only get partial relevance judgements from the strategies presented above saying

$$d_i > d_j$$

we can infer that an improved retrieval function should satisfy a list of constraints of the form

$$w * \Phi(d_{i,q}) > w * \Phi(d_{j,q}).$$

As stated in the previous section, our goal is to minimise the number of inversions  $Q$  between our ranking and the optimal ranking. For a linear ranking function like the one used here, this goal is equivalent to finding the weight vector  $w$  that fulfils the maximum number of the above inequalities. However, this maximisation problem is NP-hard

A possibility to approximate the solution of this problem is the introduction of non-negative slack variables  $\xi_{i,j}$  that allow the violation of some constraints:

$$w * (\Phi(d_{i,q}) - \Phi(d_{j,q})) \geq 1 - \xi_{i,j}$$

One can then minimise the upper bound of the sum of all the slack variables. For a series of queries  $q_1$  to  $q_n$  producing the results  $r_1$  to  $r_n$ , our optimisation problem has then the following form:

$$\begin{array}{ll} \text{Minimise} & V(w, \xi) = \frac{1}{2}w * w + C \sum \xi_{i,j,k} \\ \text{Subject to} & \forall d_i, d_j \in r_1 : w * (\Phi(d_i, q_1) - \Phi(d_j, q_2)) \geq 1 - \xi_{i,j,1} \\ & \dots \\ & \forall d_i, d_j \in r_n : w * (\Phi(d_i, q_n) - \Phi(d_j, q_n)) \geq 1 - \xi_{i,j,n} \\ & \forall i, j, k : \xi_{i,j,k} \geq 0 \end{array}$$

$C$  is a parameter that allows trading off constraint violations versus margin maximisation. The minimisation goal  $V$  also contains the norm of the weight vector  $w$  as a means of regularisation. This ensures that the elements of  $w$  don't grow unbounded.

This optimisation problem is equivalent to that of a classification SVM on pairwise difference vectors  $\Phi(q_k, d_i) - \Phi(q_k, d_j)$ . It can be solved using decomposition algorithms similar to those used for SVM classification.

### 6.3 The mapping $\Phi$

A crucial part of the construction of the SVM is the selection of the features describing the match between document and query in  $\Phi$ . When constructing this mapping one opts for features that help discriminating groups of documents in the collection.

For instance a feature “is\_html” is not that informative, as most content on the web (and indexed by search engines) is presented in HTML.

In the following, two possibilities for the contents of  $\Phi$  that were introduced in [4] and [3] will be shown.

#### 6.3.1 Contents of $\Phi$ (Version 1)

In [4], the author selected features from the following 3 categories:

- Original rank
- Query / content match, where the match is performed on similarity of the query, abstract, URL and domain name
- Popularity-attributes, that contain features like URL length or specific country top level domains

Table 3 shows a more detailed list of these features.

Rank Features	
rank	(100 - position in the original rankin) / 100
top1	document ranked 1st
top10	document ranked in top 10
top50	document ranked in top 50

Query / Content Match Features	
query_url_cosine	similarity of URL and query terms
query_abstract_cosine	similarity of query and abstract
domain_name_in_query	is the domain name composed of query terms?

Popularity Features (~20'000)	
url_length	length of URL divided by 30
country	features for different top level domains
domain	features for specific domains
abstract_contains_home	“home” appears in URL or title
url_contains_tilde	“~” is part of the URL
url	features for several fix URLs

Table 3: Proposed features for the mapping  $\Phi$ . The number of features per group is given in parenthesis.

### 6.3.2 Tests & Results

To test the learned ranking function, Thorsten Joachims [5] implemented a meta-search engine that sends queries to Google, MSNSearch (today called “Live Search”) and Toprank (a meta-search engine) and displays a composed result. After generating training data the learned ranking function was then compared to the original rankings of Google, MSNSearch and Toprank.

To compare two rankings at a time, the result page presented contains a mix of documents from the learned function and one of the 3 original rankings of the engines mentioned above. The mix is compiled such that the top n links from the mix contain  $k_a$  links from the original ranking and  $k_b$  from the learned ranking function with  $|k_a - k_b| \leq 1$ . This means if a user scans the results from top to bottom, he has at any point in time seen almost the same number of results from each ranking. If he then clicks on significantly more links from one of the two rankings it can be concluded that this ranking contains more relevant links.

A test with a group of students from the University of Dortmund showed that they preferred the links from the learned ranking (see table 4). Since they did not know they were presented such a mixed ranking, this shows that for this group the learned retrieval function is better than all the 3 original rankings.

	More Clicks on Learned	Less Clicks on Learned	Tie	No Clicks	Total
Learned vs. Google	29	13	27	19	88
Learned vs. MSN	18	4	7	11	40
Learned vs. Toprank	21	9	11	11	52

Table 4: Pairwise comparison of the learned ranking function with Google, MSNSearch and Toprank.

### 6.3.3 Analysis of the Learned Function

A look at the learned weights for the features in  $\Phi$  gives insight on how the learned retrieval function works. Most positive weight is given to the features that compare the query with the abstract or URL and to those stating that the document had a high rank in the original ranking. Large negative weight is given to documents with long URLs or those that do not show up in the first place or on a top10 position in the original ranking.

The distribution of these weights seems reasonable and makes sense intuitively.

### 6.3.4 Contents of $\Phi$ (Version 2)

In the previous sections,  $\Phi$  was composed of properties that Joachims selected by his own intuition. In [3], a more general idea to compose  $\Phi$  is presented. There are now two kinds of features:

- $\phi_{rank}$ : Rank features that weigh the relevance of the ranking in the original result.
- $\phi_{term}$ : Term/document features that will be generated for each (term, document) pair, indicating the relevance of a given document to a query term.

For a document collection  $D = \{d_1, \dots, d_M\}$  containing  $M$  documents, a query  $q$  containing  $N$  terms  $(t_1, \dots, t_N)$  and the initial ranking  $r_0$  for  $D$  according to  $q$  the mapping  $\Phi$  is composed as follows:

$$\Phi(d, q) = \begin{pmatrix} \phi_{rank}(d, q) \\ \phi_{terms}(d, q) \end{pmatrix}$$

$$\phi_{rank}(d, q) = \begin{pmatrix} \mathbf{1}(\text{Rank}(d \text{ in } r_0(d, q)) \leq 1) \\ \vdots \\ \mathbf{1}(\text{Rank}(d \text{ in } r_0(d, q)) \leq 10) \\ \mathbf{1}(\text{Rank}(d \text{ in } r_0(d, q)) \leq 15) \\ \vdots \\ \mathbf{1}(\text{Rank}(d \text{ in } r_0(d, q)) \leq 100) \end{pmatrix}$$

$$\phi_{terms}(d, q) = \begin{pmatrix} \mathbf{1}(d = d_1 \wedge t_1 \in q) \\ \vdots \\ \mathbf{1}(d = d_M \wedge t_N \in q) \end{pmatrix}$$

$\mathbf{1}$  is the indicator function.

There were 28 rank features selected that indicate whether a document was ranked above a certain position in the original ranking.

For example, if a document  $d$  is ranked 4th for a query  $q$  then  $\phi_{rank}(d, q) = (0, 0, 0, 1, \dots, 1)^T$ .

As one term/document feature for each possible (term, document) pair will be generated, there will be an extremely large number of such features. However, most of them will be set to 0, which makes the problem never the less well suited for the machine learning approach.

To increase the influence of a high rank in the original ranking, minimum weight constraints for the rank features were introduced. They were chosen such that the higher a document was in the original ranking the larger its minimum weight. The practical reason to do so is that the generated preferences from most of the previously presented strategies tend to reverse the original ranking. This effect can be limited by choosing the minimum weights accordingly.

### 6.3.5 Tests & Results

The search engine of the Cornell University Library (CUL) was used to test the learned ranking with query chains against the one without them and the original ranking of the engine. Again, the user was presented with a mix of either the original ranking and the query chains ranking, or the query chains ranking and the learned one without chains. As table 5 shows, the users preferred the results of the query chain ranking over the other two rankings. This demonstrates that analysing information on query chains brings measurable performance improvements for search engines.

	More Clicks on QC	Less Clicks on QC	Indifferent
With QC vs. Built-in	392 (32%)	239 (20%)	579 (47%)
With QC vs. No QC	211 (17%)	160 (13%)	855 (70%)

Table 5: Pairwise comparison of the learned ranking with query chains, the learned ranking without chains and the built-in ranking function of the CUL search engine.

The authors state that current (2005) SVM implementations are at their limits with this approach because of the large number of constraints generated. However, they think there is room for improvement, e.g. by using an incremental optimising approach.

## 7 Evaluating the Robustness of the Learning Method

This section explores the question whether the ranking functions learned with the presented approaches are robust to noise in user behaviour or to word ambiguity in the document collection. To find out, tests are needed and these take a long time and many users. Automated tests with simulated users would allow speeding up this process enormously. In [6], the authors have come up with methods to generate a document collection with different levels of word ambiguity and a model for user behaviour that will be presented in the following.

While such a test setup is clearly a simplification of the real world, it also gives more insight, as the relevance of each document to a query is known.

### 7.1 Document Generation

First a set of words is chosen. These are then used for defining a set of topics, where a topic is composed by randomly picked words. Some topics contain words that are more frequent than others.

The documents are then generated from the words in such a way that they are relevant to some topics. Certain topics will occur on more documents than others, which makes them more popular in the collection.

### 7.2 User Model

A user is modelled to have a level of patience and a relevance threshold. The patience determines how many results a user will view before giving up. It will decrease proportionally to the relevance of the documents the user looks at. If he sees lots of irrelevant results he will give up faster.

With each document the user associates an observed relevance that is drawn from a distribution of the real relevance of that document to the query. If this value is larger than the user's relevance threshold, he will click on it. After the click the user sees the real relevance of the document. If it is of maximum relevance the query is answered. Else he will return to the results and continue

looking through them until his patience runs out.

If his patience does run out, the user will issue a new query with a 50% probability. This then produces query chains.

### 7.3 Experiments

The following subsections describe the results of iterated learning experiments with different levels of noise in user behaviour and word ambiguity.

#### 7.3.1 Accuracy of Relevance Estimates

One parameter in the user model is the observed relevance, which is the estimated relevance of a document judging from the information presented on the results page.

Testing several levels of noise in the distribution the observed relevance is drawn from, it was found that the more noise the smaller were the learning improvements. However, the improvements are still visible even at the maximum noise level, where 48% of all preferences generated from the simulated user behaviour are wrong.

#### 7.3.2 Topic and Word Ambiguity

A similar picture is drawn by the tests with word ambiguity. By generating the documents and topics it is possible to define the level of ambiguity.

With an increasing level of ambiguity the learning performance decreases, but even in the highest ambiguity setting tested the learning algorithm still produced good results.

#### 7.3.3 User Trust in the Ranking

It has been stated before that users trust search engines to rank the most relevant documents at the top.

To experiment with higher trust into the presented ranking the observed relevance was increased proportionally to the inverse of the rank of each result. However, these different levels of trust did not show any lasting effect over several learning iterations. From this it was concluded that in spite of the strong bias for the presented ranking, clickthrough feedback still provides useful training data.

#### 7.3.4 Number of Results Looked at

During the experiments it came up that users only look at very few of the search results. To find out whether the number of viewed results has an influence on the effectiveness of the learning, different levels of user patience were tested.

There seems to be no influence on the first iterations, but during the later cycles the learning improvements taper out faster if the users view fewer results.

#### 7.3.5 Query Reformulations

The impact of the probability to reformulate a query was visible but smaller than what could be expected from the work in [3]. It is assumed that this originates from later queries not being identically distributed to the earlier ones as simulated here.

The hypothesis is that later queries are better. This was indeed affirmed by human judges for the data collected during the eye tracking study when comparing the top results from two queries in a chain.

## 7.4 Conclusions

The presented learning method is robust to noise in user behaviour in a number of different document collections with varying word ambiguity.

The approach of modelling document collections and the user behaviour allows fast exploration of the different parameters that influence the ability to learn an improved ranking function.

# 8 Collecting Unbiased Preferences from Clickthrough Data

A problem of using clickthrough data for implicit feedback is the fact that it contains noise. This can be explained by different users probably having different understandings of the relevance of a document to a query. Another source for noise would be “click spam” where users intentionally click on some link to abuse the ranking system. Their goal is for instance to get a higher ranking for their own site. This section shows a possibility to collect cleaner training data from users’ clicks.

The idea presented in [7] is to reduce the bias of presentation that makes users click more often on higher ranked documents. To achieve this, the presentation of the search results is slightly modified while having as little effect on the quality of the result as possible. An algorithm called FairPairs will be shown for that purpose.

## 8.1 Presentation Bias

Interpreting clicks as relative feedback about pairs of results as done in the methods previously described, avoids the presentation bias problem.

However, the preferences generated by these strategies almost always oppose the presented ordering. This means that reversing the initial ranking can trivially satisfy them. To prevent that, the place in the initial ranking has a reasonably high weight in the SVM approach described above.

## 8.2 FairPairs

FairPairs is a method to collect presentation bias free preference statements from clicks. The idea is to switch some pairs of adjacent results and then record the clicks. As two adjacent results have very similar relevance the user is unlikely to notice such a switch. The goal is for each pair of links to present them to different users, in normal and in switched order.

By comparing the number of clicks when a document is in the top position in the pair versus when it is the bottom one, a bias-free preference statement can be generated.

It can be shown that using FairPairs the learned ranking will converge to an ideal ranking, if one exists (see [7] for details). This point is not true for the several previously presented strategies to generate preferences (like “Click > Skip Above”). Those tend to reverse the original ranking, which does not automatically converge to an ideal ranking.

## 8.3 Conclusions

It is unknown how FairPairs actually compares to the initial strategies with or without query chains, as there are no measurements or comparisons in that direction presented. In the given version FairPairs does not exploit query chains. The authors state it would be an interesting challenge trying to extend it into that direction.

## 9 Wrap Up

The approach presented in this report was a new method to elicit implicit feedback from the vast resource of search engine log files and then train a support vector machine to learn an improved ranking function.

This idea underwent several refinements and improvements.

A number of strategies to extract preference constraints from the observed clicks were proposed and tested. The results were confirmed in a user study that incorporated eye tracking. To get even more information out of the logs, the concept of query chains was successfully established. Using a generated document collection and simulated user behaviour, the presented approach proved robust against noise in the training data and different levels of term ambiguity.

And last but not least, the introduction of FairPairs made it possible to collect implicit relevance judgements without presentation bias.

Points still open are:

- the influence of “click spam” on the learning algorithm
- the question if FairPairs can be extended to exploit query chains
- the scalability of this approach to large scale web search
- the extension of this approach to personalised ranking functions
- how the different strategies can/should be combined

## 10 References

- [1] Text Retrieval Conference TREC: <http://trec.nist.gov/>
- [2] T. Joachims, L. Granka, Bing Pan, H. Hembrooke, F. Radlinski, G. Gay. *Evaluating the Accuracy of Implicit Feedback From Clicks and Query Reformulations in Web Search* (to appear)
- [3] F. Radlinski and T. Joachims. *Query Chains: Learning to Rank From Implicit Feedback* (2005)
- [4] Thorsten Joachims. *Optimizing Search Engines Using Clickthrough Data* (2002)
- [5] Thorsten Joachim’s Web Page: <http://www.cs.cornell.edu/People/tj/>
- [6] F. Radlinski and T. Joachims. *Evaluating the Robustness of Learning From Implicit Feedback* (2005)
- [7] F. Radlinski and T. Joachims. *Minimally Invasive Randomization for Collecting Unbiased Preferences From Clickthrough Logs* (2005)