OPTIMIZING SEARCH ENGINES USING CLICKTHROUGH DATA

Paper By: Thorsten Joachims (Cornell University)
Presented By: Roy Levin (Technion)
Outline

- The idea
- The model
- Learning a ranking function
- Experimental results
- Conclusion
The idea is to use **clickthrough data** for optimizing the retrieval quality of search engines.

- Such clickthrough data is available in abundance and can be recorded at very low cost to the query-log of search engines.
  - Return list of proxy urls instead of actual urls.
- Retrieval functions are learned using an SVM approach.
What Can Clickthrough Data Help?

- **Problem 1:**
  - How to measure the retrieval quality of a search engine?
  - How to compare the performance between two search engines?
    - I.e. which search engine provides better results?
    - Google or MSNSearch?
    - Users are only rarely willing to give explicit feedback.

- **Problem 2:**
  - How to improve the ranking function of search engines?
  - Can we learn something like “for query $q$, document $a$ should be ranked higher than document $b$”?
Previous Approaches

- Learning retrieval functions from examples:
  - Typically require training data generated from relevance judgments by experts
  - This makes them difficult and **expensive** to apply
Clickthrough Data in Search Engines

- A triplets \((q, r, c)\) represents clickthrough data
  - Query, \(q\)
  - Ranking presented to the user, \(r\)
  - Set of links the user clicked on, \(c\)

- Users do not click on links at random, they make a (somewhat) informed choice.

- While clickthrough data is typically noisy and not “perfect” relevance judgments, they are likely to convey some information.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kernel Machines</td>
<td><a href="http://svm.first.gmd.de/">http://svm.first.gmd.de/</a></td>
</tr>
<tr>
<td>4</td>
<td>An Introduction to SVMs</td>
<td><a href="http://www.support-vector.net/">http://www.support-vector.net/</a></td>
</tr>
<tr>
<td>6</td>
<td>Archives</td>
<td><a href="http://www.jiscmail.ac.uk/lists/support-vector-machine.html">http://www.jiscmail.ac.uk/lists/support-vector-machine.html</a></td>
</tr>
<tr>
<td>8</td>
<td>Royal Holloway Support Vector Machine</td>
<td><a href="http://svm.dcs.rhbnc.ac.uk/">http://svm.dcs.rhbnc.ac.uk/</a></td>
</tr>
<tr>
<td>9</td>
<td>SVM - The Software</td>
<td><a href="http://www.support-vector.net/software.html">http://www.support-vector.net/software.html</a></td>
</tr>
</tbody>
</table>
There are strong dependencies between the three parts of \((q, r, c)\)

- **Good news:**
  - Users are more likely to click on a link relevant to \(q\)

- **Bad news:**
  - Clicks also depend on the presented ranking \(r\)

- The presented ranking \(r\), depends on the query \(q\)
  - Determined by the retrieval function implemented in the search engine

- It is necessary to consider and model the dependencies of \(c\) on \(q\) and \(r\) appropriately
Empirical experiments with 3 different retrieval strategies averaged over ~1400 queries show:

- The retrieval functions are substantially different in their ranking quality based on subjective judgments.
- However, the observed average clickrank is not very different.
  - Users typically scan only the first 10 links.
  - Hence clicking on a link cannot be interpreted as a relevance judgment on an absolute scale.
What can we assume?

- Assumption:
  - user scanned ranking from top to bottom
- In our “SVM” query example:
  - user must have observed link 2 before clicking on 3, making a decision to not click on it
  - plausible to infer that link 3 is more relevant than link 2 with probability higher than random
  - similarly, 7 is more relevant than links 2, 4, 5, 6
What can we assume? (continued)

- The information obtained from user on $r^*$ (ranking) is:
  - $\text{link}_3 <_{r^*} \text{link}_2$
  - $\text{link}_7 <_{r^*} \text{link}_2$
  - $\text{link}_7 <_{r^*} \text{link}_4$
  - $\text{link}_7 <_{r^*} \text{link}_5$
  - $\text{link}_7 <_{r^*} \text{link}_6$

- Generally:
  - $\{\text{link}_i <_{r^*} \text{link}_j, \forall (i, j) | 1 \leq j < i, i \in C, j \notin C\}$

- Unfortunately, this type of feedback is not suitable for standard machine learning algorithms.
Outline

- The idea
- The model
- Learning a ranking function
- Experimental results
- Conclusion
Traditional IR performance measures

- Why not use traditional performance measures?
  - i.e. precision and recall?
- Not appropriate for multi-grade relevance
- The approach here is different
  - basically all documents can be said to be relevant, it's only a matter of ranking
  - i.e. saying that a document is ranked very high is the same as saying it's irrelevant
Framework for Learning Retrieval Functions

- A query $q$ containing all info related to the search including user properties and etc.
- A document collection $D=\{d_1, d_2, \ldots, d_m\}$
- An optimal ordering function
  - binary relation $r^* \subset D \times D$
- Operational retrieval function $f$
  - ordering function $r_{f(q)} \subset D \times D$
- $f$ is evaluated in terms of how $r_{f(q)}$ is close to $r^*$
  - what does close here mean?
Properties of the Ordering Relation

- An ordering relation $r$ is irreflexive
  - i.e. $(1,1) \notin r$

- An ordering relation $r$ is asymmetric
  - i.e. $(3,2) \in r \Rightarrow (2,3) \notin r$

- An ordering relation $r$ is also negatively transitive
  - i.e. $(3,1) \notin r, (7,3) \notin r \Rightarrow (7,1) \notin r$

- For simplicity (for now) lets assume that $r^*$ and $r_{f(q)}$ provide strict orderings
  - i.e. $(a,b) \in D \times D \Rightarrow (a,b) \in r \lor (b,a) \in r$
Performance Measure Using Kendall's Tau

- Most frequently used measure in statistics
  - Used for comparing the ordinal correlation of two random variables
- Given two ordering relations \( r_a \) and \( r_b \)
  - \( P = \# \) of concordant pairs
    - concordant on \( \{2,3\} \) when \( 3 \leq r_a 2 \) iff \( 3 \leq r_b 2 \)
  - \( Q = \# \) of discordant pairs (inversions)
- \([-1,1]\) \( \exists \tau(r_a, r_b) = \frac{P - Q}{P + Q} = \frac{(m^2) - Q}{m^2} - Q = \frac{m!}{(m/2)!} \frac{2Q}{m^2} = 1 - \frac{2Q}{m^2} \)
Performance Measure Using Kendall's Tau
(an example)

- For example:
  - \( d_1 <_r a d_2 <_r a d_3 <_r a d_4 <_r a d_5 \)
  - \( d_3 <_r b d_2 <_r b d_1 <_r b d_4 <_r b d_5 \)

- \#discordant-pairs=3
  - \( \{d_1, d_2\}, \{d_1, d_3\}, \{d_2, d_3\} \)

- \#concordant-pairs=7

- \( \tau(r_a, r_b) = 1 - \frac{2 \cdot 3}{\binom{5}{2}} = 1 - \frac{6}{\frac{5!}{3!2!}} = 1 - \frac{6}{10} = \frac{2}{5} = 0.4 \)
Why is Kendell’s tau an appropriate measure for IR?

- Depends only on Q for a fixed collection
- In a binary relevance scale:
  - Maximizing the tau is equivalent to minimizing the average rank of the relevant documents
  - Every relevant document receives a lower rank than non-relevant documents
The Axioms of Kemeny and Snell for Strict Orderings

- Taken as a distance measure, Q fulfills the axioms of Kemeny and Snell for strict orderings
  - Based on pairwise comparisons of the form:
    - object \( b \) is preferred to object \( a \), object \( a \) is preferred to object \( d \), . . . Etc
  - Unique distance between any pair of rankings exists
  - The median ranking problem is then to find, for a given set of rankings, a ranking \( X \) for which:
    - the total distance from \( X \) to this set is minimized
A fixed but unknown distribution $Pr(q, r^*)$ of queries and target rankings on $D$

The goal is to learn a retrieval function $f(q)$ for which the expected Kendall’s tau is maximal

$$\tau_p(f) = \int \tau(r_{f(q)}, r^*)d Pr(q, r^*)$$

The question remains whether it is possible to design learning methods that optimize $tau_p$?
Outline

- The idea
- The model
- Learning a ranking function
- Experimental results
- Conclusion
Learning a Ranking Function

- **Training sample S**
  \[ S = \{(q_1, r_1^*), (q_2, r_2^*), \ldots, (q_n, r_n^*)\} \]

- The learner \( L \) will select a ranking function \( f \) from a family of ranking functions \( F \) that maximize \( \tau \)-s
  \[ \tau_s(f) = \frac{1}{n} \sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*) \]
Learning a Ranking Function (continued)

- Is it possible to design an algorithm and a family of ranking functions $F$ so that:
  - finding the function $f$ in $F$ for maximizing $\tau$-s is efficient
  - this function generalizes well beyond the training data
Consider the class of linear ranking functions

- \((d_i, d_j) \in f_w(q) \iff \vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j)\)
- \(\vec{w}\) is a weight vector adjusted by learning
- \(\Phi(q, d_i)\) vector of matching features between \(q\) and \(d_i\)
  - As done in classic IR – matching words, page rank of \(d_i\) etc.
- Find weight vector that maximizes the average \(\tau\)
The Weight Vector

- Foreach \( \vec{w} \) ordered points \( \Phi(q, d_i) \) by their projection on \( w \)
- Foreach \( q \) we seek \( \vec{w} \) that orders points correctly
- For entire dataset we seek vector that
  - minimize number of discordant pairs

Example of how two weight vectors \( w_1 \) and \( w_2 \) rank four points
Maximizing tau-s is equivalent to minimizing the number Q of discordant pairs.

This is equivalent to finding $\vec{w}$ that maximizes $\#$ of the following inequalities that are fulfilled:

- $\forall (d_i, d_j) \in r_1^* : \vec{w} \cdot \Phi(q_1, d_i) > \vec{w} \cdot \Phi(q_1, d_j)$
- ...
- $\forall (d_i, d_j) \in r_n^* : \vec{w} \cdot \Phi(q_n, d_i) > \vec{w} \cdot \Phi(q_n, d_j)$

Unfortunately, this problem is NP-Hard.
The Categorization SVM

Learning ⇒ Optimization problem

Minimize: \[
\frac{1}{2} \vec{w} \cdot \vec{w}
\]
Subject to:
\[
\forall i: y_i = 1 \Rightarrow \vec{w} \cdot \vec{\Omega}_i + b \geq 1
\]
\[
\forall i: y_i = -1 \Rightarrow \vec{w} \cdot \vec{\Omega}_i + b \leq -1
\]

Learning ⇒ Optimization problem (with error)

Minimize: \[
\frac{1}{2} \vec{w} \cdot \vec{w} + C \sum \xi_i
\]
Subject to:
\[
\forall i: y_i = 1 \Rightarrow \vec{w} \cdot \vec{\Omega}_i + b \geq 1 - \xi_i
\]
\[
\forall i: y_i = -1 \Rightarrow \vec{w} \cdot \vec{\Omega}_i + b \leq -1 + \xi_i
\]

\[y_i = +1 (-1) \text{ if } \Omega_i \text{ is in class } + (-)\]
The ranking SVM algorithm

Introduce slack variables $\xi_{i,j,k}$

Minimize

$$V(w, \xi) = \frac{1}{2} w \cdot w + C \sum \xi_{i,j,k}$$

Subject to

$$w(\Phi(q_k, d_i) - \Phi(q_k, d_j)) \geq 1 - \xi_{i,j,k}$$

$$\xi_{i,j,k} \geq 0$$

The optimization problem is equivalent to that of a classification SVM on pairwise difference vectors.

Due to this similarity:

The problem can be solved using decomposition algorithms similar to those used for SVM classification (SVM light).
Clickthrough logs are the source of training data

- Hence the full target ranking $r^*$ for $q$ is not available
- $r' \subset r^*$ is available in the logfile

Adapt the Ranking SVM to the case of such partial data by is straightforward

- Just replacing $r^*$ with $r'$
Outline

- The idea
- The model
- Learning a ranking function
- Experimental results
- Conclusion
Experiment Setup: Meta Search

Search query $f_w$ leads to Combined results and Clickthrough data.
Combined Rankings

- Two separate rankings $A$ and $B$
- Combined into a single ranking $C$
- For any top $l$ links in $C$
  - the top $k_a$ links from $A$ and the top $k_b$ links from $B$ are combined with $|k_a - k_b| \leq 1$
- If the user scans the links of $C$ from top to bottom
  - at any point he has seen almost equally many links from the top of $A$ as from the top of $B$
# Combined Rankings (example)

## Ranking A:
1. Kernel Machines  
   [http://svm.first.gmd.de/](http://svm.first.gmd.de/)
2. SVM-Light Support Vector Machine  
3. Support Vector Machine and Kernel Methods, References  
   [http://www.support--vector.net/SVMRefs.html](http://www.support--vector.net/SVMRefs.html)
4. Lucent Technologies: SVM demo applet  
   [http://www.support-vector.net/SVTSVMsvt.html](http://www.support-vector.net/SVTSVMsvt.html)
5. Royal Holloway Support Vector Machine  
   [http://svm.dcs.rhbnec.ac.uk/](http://svm.dcs.rhbnec.ac.uk/)
   [http://www.support-vector.net/software.html](http://www.support-vector.net/software.html)
7. Support Vector Machine - Tutorial  
   [http://www.support-vector.net/tutorial.html](http://www.support-vector.net/tutorial.html)
8. Support Vector Machine  

## Ranking B:
1. Kernel Machines  
   [http://svm.first.gmd.de/](http://svm.first.gmd.de/)
2. Support Vector Machine  
3. An Introduction to Support Vector Machines  
4. Archives of SUPPORT-VECTOR-MACHINES  
   [http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html](http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html)
5. SVM-Light Support Vector Machine  
   [http://www.support-vector.net/software.html](http://www.support-vector.net/software.html)
7. Lagrangian Support Vector Machine Home Page  
8. A Support ... - Bennett, Blue (ResearchIndex)  
   [http://citeseer.../bennett97support.html](http://citeseer.../bennett97support.html)

## Combined Results:
1. Kernel Machines  
   [http://svm.first.gmd.de/](http://svm.first.gmd.de/)
2. Support Vector Machine  
3. SVM-Light Support Vector Machine  
4. An Introduction to Support Vector Machines  
5. Support Vector Machine and Kernel Methods, References  
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK  
   [http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html](http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html)
7. Lucent Technologies: SVM demo applet  
8. Royal Holloway Support Vector Machine  
   [http://svm.dcs.rhbnec.ac.uk/](http://svm.dcs.rhbnec.ac.uk/)
   [http://www.support--vector.net/software.html](http://www.support--vector.net/software.html)
10. Lagrangian Support Vector Machine Home Page  
Offline Experiment

- **Purpose:**
  - verify that the Ranking SVM can indeed learn based on partial feedback from clickthrough data.
  - Striver displayed the combined results of Google and MSNSearch
    - data was collected by a single user during use in a single month
    - all clickthrough triplets were recorded
      - resulted in 112 queries with a non-empty set of clicks
Framework for Offline Experiments

- Learning a retrieval function using the Ranking SVM requires:
  - designing a suitable feature mapping $\phi(q, d)$ to describe the match between query $q$ and document $d$
- Using only simplistic, easy to implement features
- Better feature mappings is likely to yield better results
Feature Mapping Type 1

- **Rank in other search engines:**
  - **rank \( X \):** \( \min \left( 100 - \frac{\text{combined rank} (x)}{100}, 0 \right) \)
  - **top1\(_X\):** \( \text{combined rank} (x) = 1 \) (binary \{0, 1\})
  - **top10\(_X\):** \( \text{combined rank} (x) \leq 10 \) (binary \{0, 1\})
  - **top50\(_X\):** \( \text{combined rank} (x) \leq 50 \) (binary \{0, 1\})
  - **top1count\(_X\):** ranked \#1 in \( X \) of the 5 search engines
  - **top10count\(_X\):** top 10 rank in \( X \) of 5 search engines
  - **top50count\(_X\):** top 50 rank in \( X \) of 5 search engines
Feature Mapping Type 2

- **Query/Content Match:**
  - **query URL cosine:**
    - cosine between URL-words and query \((range [0, 1])\)
  - **query abstract cosine:**
    - cosine between title-words and query \((range [0, 1])\)
  - **domain name in query:**
    - query contains domain-name from URL \((binary \{0, 1\})\)
Feature Mapping Type 3

- **Popularity-Attributes:**
  - **url length:**
    - length of URL in characters divided by 30
  - **country X:**
    - country code X of URL (binary attribute for each country code)
  - **domain X:**
    - domain X of URL (binary attribute for each domain name)
  - **abstract contains home:**
    - word “home” appears in URL or title (binary attribute)
  - **url contains title:**
    - URL contains “~” (binary attribute)
  - **url X:**
    - URL contains X as an atom (binary attribute)
Offline Experimental Results
Interactive Online Experiment

- **Purpose:**
  - Verify that the learned retrieval function improves retrieval quality as desired
  - Striver search engine was made available to a group of approximately 20 users
  - After ~month the system had collected 260 training queries (with at least one click)
Interactive Online Experimental Results

<table>
<thead>
<tr>
<th>Comparison</th>
<th>more clicks on learned</th>
<th>less clicks on learned</th>
<th>tie (with clicks)</th>
<th>no clicks</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learned vs. Google</td>
<td>29</td>
<td>13</td>
<td>27</td>
<td>19</td>
<td>88</td>
</tr>
<tr>
<td>Learned vs. MSNSearch</td>
<td>18</td>
<td>4</td>
<td>7</td>
<td>11</td>
<td>40</td>
</tr>
<tr>
<td>Learned vs. Toprank</td>
<td>21</td>
<td>9</td>
<td>11</td>
<td>11</td>
<td>52</td>
</tr>
</tbody>
</table>

- how many queries users click on more/less links from the top of the learned retrieval function

- Comparison with Google
  - on 29 queries users click on more links from learned
  - on 13 queries they click on more links from Google
  - on 27 queries they click on equal number
  - on 19 queries they click on neither
## Analysis of the Learned Function

<table>
<thead>
<tr>
<th>weight</th>
<th>feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60</td>
<td>query_abstract_cosine</td>
</tr>
<tr>
<td>0.48</td>
<td>top10_google</td>
</tr>
<tr>
<td>0.24</td>
<td>query_url_cosine</td>
</tr>
<tr>
<td>0.24</td>
<td>top1count_1</td>
</tr>
<tr>
<td>0.24</td>
<td>top10_msnsearch</td>
</tr>
<tr>
<td>0.22</td>
<td>host_citesee</td>
</tr>
<tr>
<td>0.21</td>
<td>domain_nec</td>
</tr>
<tr>
<td>0.19</td>
<td>top10count_3</td>
</tr>
<tr>
<td>0.17</td>
<td>top1_google</td>
</tr>
<tr>
<td>0.17</td>
<td>country_de</td>
</tr>
<tr>
<td>...</td>
<td>abstract_contains_home</td>
</tr>
<tr>
<td>0.16</td>
<td>top1_hotbot</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0.14</td>
<td>domain_name_in_query</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>-0.13</td>
<td>domain_tu-bs</td>
</tr>
<tr>
<td>-0.15</td>
<td>country_fin</td>
</tr>
<tr>
<td>-0.16</td>
<td>top50count_4</td>
</tr>
<tr>
<td>-0.17</td>
<td>url_length</td>
</tr>
<tr>
<td>-0.32</td>
<td>top10count_0</td>
</tr>
<tr>
<td>-0.38</td>
<td>top1count_0</td>
</tr>
</tbody>
</table>
Conclusions

- **Pros:**
  - The feasibility of using clickthrough data in evaluating retrieval performance has been verified.
  - Machine learning techniques can improve retrieval substantially by:
    - tailoring the retrieval function to small and homogenous groups (or even individuals) without prohibitive costs.

- **Cons:**
  - Links that are relevant but ranked very low still remain invisible.
  - Approaches have not been justified in large scales.
    - Whether or not the techniques are workable in real cases is still uncertain.