Enhancing Web Search by Promoting Multiple Engine Use

Ryen W. White, Matthew Richardson, Mikhail Bilenko
Microsoft Research

Allison Heath
Rice University
User Loyalty

- Users are generally loyal to one engine
  - Even when engine switching cost is low, and even when they are unhappy with search results
- Change can be inconvenient, users may be unaware of other engines

- A given search engine performs well for some queries and poorly for others
  - Excessive loyalty can hinder search effectiveness
Our Goal

- Support engine switching by recommending the most effective search engine for a given query
- Users can use their default but have another search engine suggested if it has better results
Overview

- Switching support vs. meta-search
- Characterizing current search engine switching
- Supporting additional switching
- Evaluating switching support
- Conclusions and implications
Relationship to Meta-Search

Meta-search:
- Merges search results
- Requires change in default engine (< 1% share)
- Obliterates benefits from source engine UX investments
- Hurts source engine brand awareness

We let users keep their default engine and suggest an alternative engine if we estimate it performs better for the current query.
Does switching help users?
A Case for Switching

- Pursued statistical clues on switching behavior
- Aims:
  - Characterize switching
  - Understand if switching would benefit users

- Extracted millions of search sessions from search logs
  - Began with query to Google, Yahoo!, or Live
  - Ended with 30 minutes of user inactivity
6.8% of sessions had switch
12% of sessions with > 1 query had switch

Three classes of switching behavior:

- **Within-session** (33.4% users)
- **Between-session** (13.2% users) – Switch for different sessions (engine task suitability?)
- **Long-term** (7.6% users) – Defect with no return

Most users are still loyal to a single engine
Potential Benefit of Switching

- Quantify benefit of multiple engine use
  - Important as users must benefit from switch

- Studied search sessions from search logs
- Evaluated engine performance with:
  - Normalized Discounted Cumulative Gain (NDCG)
  - Search result click-through rate
- 5K query test set, Goo/Yah/Live query freq. ≥ 5
Potential Benefit of Switching (cont.)

Six-level relevance judgments, e.g.,

\[ q = [\text{black diamond carabiners}] \]

<table>
<thead>
<tr>
<th>URL</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.bdel.com/gear">www.bdel.com/gear</a></td>
<td>Perfect</td>
</tr>
<tr>
<td><a href="http://www.climbing.com/Reviews/biners/Black_Diamond.html">www.climbing.com/Reviews/biners/Black_Diamond.html</a></td>
<td>Excellent</td>
</tr>
<tr>
<td><a href="http://www.climbinggear.com/products/listing/item7588.asp">www.climbinggear.com/products/listing/item7588.asp</a></td>
<td>Good</td>
</tr>
<tr>
<td><a href="http://www.rei.com/product/471041">www.rei.com/product/471041</a></td>
<td>Good</td>
</tr>
<tr>
<td><a href="http://www.nextag.com/BLACK-DIAMOND/">www.nextag.com/BLACK-DIAMOND/</a></td>
<td>Fair</td>
</tr>
<tr>
<td><a href="http://www.blackdiamondranch.com/">www.blackdiamondranch.com/</a></td>
<td>Bad</td>
</tr>
</tbody>
</table>

\[
\text{NDCG}(i) = N_i \sum_i \frac{2^{r(i)} - 1}{\log (1 + i)}
\]

We use NDCG at rank 3
**Computed same stats on all instances of the queries in logs (not just unique queries)**

**For around 50% of queries there was a different engine with better relevance or CTR**

**Engine choice for each query is important**

### Number (%) of 5K unique queries that each engine is best

<table>
<thead>
<tr>
<th>Search engine</th>
<th>Relevance (NDCG)</th>
<th>Result click-through rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>952 (19.3%)</td>
<td>2,777 (56.4%)</td>
</tr>
<tr>
<td>Y</td>
<td>1,136 (23.1%)</td>
<td>1,226 (24.9%)</td>
</tr>
<tr>
<td>Z</td>
<td>789 (16.1%)</td>
<td>892 (18.1%)</td>
</tr>
<tr>
<td>No difference</td>
<td>2,044 (41.5%)</td>
<td>26 (0.6%)</td>
</tr>
</tbody>
</table>
Can we support switching?
Supporting Switching

- Users may benefit from recommendations
  - Find a better engine for their query
- Model comparison as binary classification
  - Closely mirrors the switching decision task
- Actual switch utility depends on cost/benefit
  - Using a quality margin can help with this
  - Quality difference must be $\geq$ margin

- Used a maximum-margin averaged perceptron
Switching as Classification

Query $q$
Result page (origin) $R$
Result page (target) $R'$

Human-judged result set with $k$ ordered URL-judgment pairs $R^* = \{(d_1, s_1), \ldots, (d_k, s_k)\}$

Utility of each engine for each query is represented by the NDCG score $U(R) = NDCG_{R^*}(R)$ $U(R') = NDCG_{R^*}(R')$

Provide switching support if utility higher by at least some margin...

Dataset of queries $Q = \{(q, R, R', R^*)\}$
yields a set of training instances $D = \{(x, y)\}$

Where each instance $x = f(q, R, R')$

\[ y = 1 \text{ iff } NDCG_{R^*}(R') \geq NDCG_{R^*}(R) + \text{margin} \]
Classifier Features

- Classifier must recommend engine in real-time
  - Feature generator needs to be fast
  - Derive features from result pages and query-result associations

Features:
- Features from result pages
- Features from the query
- Features from the query-result page match
Result Page Features - e.g.,
10 binary features indicating whether there are 1-10 results
Number of results
For each title and snippet:
   # of characters
   # of words
   # of HTML tags
   # of “…” (indicate skipped text in snippet)
   # of “.” (indicates sentence boundary in snippet)
   # of characters in URL
   # of characters in domain (e.g., “apple.com”)
   # of characters in URL path (e.g., “download/quicktime.html”)
   # of characters in URL parameters (e.g., “?uid=45&p=2”)
3 binary features: URL starts with “http”, “ftp”, or “https”
5 binary features: URL ends with “html”, “aspx”, “php”, “htm”
9 binary features: .com, .net, .org, .edu, .gov, .info, .tv, .biz, .uk
   # of “/” in URL path (i.e., depth of the path)
   # of “&” in URL path (i.e., number of parameters)
   # of “=” in URL path (i.e., number of parameters)
   # of matching documents (e.g., “results 1-10 of 2375”)
Query Features - e.g.,

- # of characters in query
- # of words in query
- # of stop words (a, an, the, ...)
- 8 binary features: Is \( i \)th query token a stopword
- 8 features: word lengths (# chars) from smallest to largest
- 8 features: word lengths ordered from largest to smallest
- Average word length

Match Features - e.g.,

For each text type (title, snippet, URL):

- # of results where the text contains the exact query
- # of top-1, top-2, top-3 results containing query
- # of query bigrams in the top-1, top-2, top-3, top-10 results
- # of domains containing the query in the top-1, top-2, top-3
Query Processing

1. Query
2. Search Engines
3. Feature Extractor

Search Engine Federator

Result sets

Features

Classifier (trained offline)

Recommendation
Evaluation

- Evaluate accuracy of switching support to determine its viability

**Task:** Accurately predict when one search engine is better than another

**Ground truth:**
- Used labeled corpus of queries randomly sampled from search engine logs
- Human judges evaluated several dozen top-ranked results returned by Google, Yahoo, and Live Search
Evaluation (cont.)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of queries</td>
<td>17,111</td>
</tr>
<tr>
<td>Total number of judged pages</td>
<td>4,254,730</td>
</tr>
<tr>
<td>Total number of judged pages labeled <em>Fair</em> or higher</td>
<td>1,378,011</td>
</tr>
</tbody>
</table>

- 10-fold cross validation, 100 runs, randomized fold assignment
Evaluation (cont.)

- Trade-offs (recall, interruption, error cost)
- Low confidence threshold = more erroneous recommendations, more frequent
- Preferable to interrupt user less often, with higher accuracy
- Use P-R curves rather than single accuracy point
  - Prec. = # true positive / total # predicted positives
  - Recall = # true positives / total # true positives
- Vary the confidence threshold to get P-R curve
Precision low (~50%) at high recall levels

- Low threshold, equally accurate queries are viewed as switch-worthy

- Demonstrates the difficulty of the task
Findings – Precision/Recall

- Goal is to provide **additional value** over current search engine
- Provide accurate switching suggestions
- Infrequent user interruption, every q not needed

Summary of precision at recall=0.05.

<table>
<thead>
<tr>
<th>From</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>0.758</td>
<td>0.883</td>
</tr>
<tr>
<td>Y</td>
<td>0.811</td>
<td></td>
<td>0.816</td>
</tr>
<tr>
<td>Z</td>
<td>0.860</td>
<td>0.795</td>
<td></td>
</tr>
</tbody>
</table>

- Classifier would fire accurately for 1 query in 20
Findings – Current engine only

- Querying additional engine may add network traffic, undesirable to target engine

- Accuracy lower, but latency may be less
Findings – Feature Contribution

- All sets of features contribute to accuracy
- Features obtained from result pages seem to provide the most benefit
Conclusions and Take-away

- Demonstrated potential benefit of switching
- Described a method for automatically determining when to switch engines for a given query
- Evaluated the method and illustrated good performance, especially at usable recall

- Switching support is an important new research area that has potential to really help users
Current and Future Directions

- **User studies:**
  - **Task:** Switching based on search task rather than just search queries
  - **Interruption:** Understanding user focus of attention and willingness to be interrupted
  - **Cognitive burden** of adapting to new engine