Search Shortcuts Using Click-Through Data

Ranieri Baraglia
Raffaele Perego
Fabrizio Silvestri
ISTI-CNR Pisa (Italy)

Fidel Cacheda
Victor Carneiro
Vreixo Formoso
University of A Coruña (Spain)
Outline

- Introduction
- Search shortcuts model
- Search shortcuts algorithms
- Experiments
- Conclusions and future work
Introduction

- Give suggestions to the users is quite common in the Web: Amazon, Netflix, ...
- Also search engines provide suggestions: queries
- In all cases, the objective is drive the user towards the information bits they may need
- But, the suggestion task is complex:
  - Design effective algorithms
  - Evaluation: user studies (precision, non repeatability)
- In this work, we propose the Search Shortcuts
Introduction

- Search shortcuts:
  - Help users to rapidly satisfy their information needs by showing shortcuts (~ queries)
  - Exploit the experiences of previous users (query logs)
Introduction

- Search shortcuts:
  - Help users to rapidly satisfy their information needs by showing shortcuts (~ queries)
  - Exploit the experiences of previous users (query logs)

\[
\begin{align*}
q_1 & \rightarrow q_2 & q_3 & \rightarrow q_4 & \rightarrow \text{Doc} \\
& & \text{Try } q_4 \quad (\text{or } q_3, \text{ or } q_2) \\
& & q_1
\end{align*}
\]
Introduction

• Search shortcuts:
  – Help users to rapidly satisfy their information needs by showing shortcuts (~ queries)
  – Exploit the experiences of previous users (query logs)

• In this work we define:
  – The Search Shortcut Discovery Problem
  – An evaluation based on query logs (e.g. Microsoft 2006 RFP) => Straightforward comparison of different techniques.
Search Shortcuts Model

- A query session for a user: \( \sigma = \langle q_1 \ldots q_n \rangle \)

- Function \( c(\sigma, i) \):
  - 1 if in \( \sigma \) the user has clicked on at least a result in \( q_i \) (0 otherwise)

- A session \( \sigma \) is **satisfactory** if \( c(\sigma, n) = 1 \)

- A **k-way shortcut** is a function \( h \) that takes as argument a session and returns a set of queries (of cardinality less than \( k \)).

- Head \( (\sigma_t|) \) and tail \( (\sigma_t|t) \) of a session

- Similarity of a k-way shortcut \( h \) on a session \( (\sigma_t| \) and \( \sigma_t|t) \)
  \[
  s(\ h(s_t|, s_t|)) = \frac{\sum_{q \in h(s_t|) m=1}^{n-t} [q = (s_t|) m] f(m)}{|h(s_t|)|}
  \]
Search Shortcuts Model

- Similarity score:
  - Measures how many queries from the shortcut are in the tail of the session
  - Valid also for query suggestion (look for similar queries)
- Importance of $f(m)$ in the similarity score:
  - Give more importance to the best shortcuts

$$\sigma = q_1, q_2, q_3, q_4, q_5 \text{ and } q_6$$

$$h_1(q_1, q_2, q_3) = \{q_4, q_6, q_8\}$$

$$h_2(q_1, q_2, q_3) = \{q_4, q_5, q_8\}$$

<table>
<thead>
<tr>
<th></th>
<th>$f(m)$</th>
<th>1</th>
<th>$m$</th>
<th>$m^2$</th>
<th>$e^m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0.67</td>
<td>1.33</td>
<td>3.33</td>
<td>7.60</td>
<td></td>
</tr>
<tr>
<td>$h_2$</td>
<td>0.67</td>
<td>1.00</td>
<td>1.67</td>
<td>3.73</td>
<td></td>
</tr>
</tbody>
</table>
Search Shortcuts Algorithms

• Collaborative filtering:
  - Traditionally: recommend items to the user based on ratings (implicit or explicit) given by other users
  - Based on the rating matrix (users x items)

• Search shortcuts and collaborative filtering:
  - No rating matrix, but query log => Infer the matrix from it: users ~ sessions and items ~ queries
  - Extract the ratings:
    • Last query successful => + (unsuccessful => -)
    • Other queries => 0
  - Query logs vs. (recommender) data sets:
    • Less dense => Sparsity problem

Much more data and more dynamic
Search Shortcuts Algorithms

- Ad-hoc algorithm: find similarities among the queries and URLs from the query log
  - Matrix of queries and documents (relevants)
  - Coclustering of the matrix: clusters of related queries
  - Session inversion: infer the last (successful) query for the sessions in which a query appears
  - Recommendations:
    - Get all queries from the same cluster
    - Follow inverted index to find similar queries
    - Recommendation list: intersect the queries found for each cluster
Search Shortcuts Algorithm

- S1: apple, banana, coffee, cake, tea
- S2: apple, orange, spaghetti, raspberry
- S3: orange, raspberry, pie, coffee
Search Shortcuts Algorithm

- Apple => tea, raspberry
- Banana => tea
- Coffee => tea
- Cake => tea
- Orange => raspberry, coffee
- Spaghetti => raspberry
- Raspberry => coffee
- Pie => coffee
Search Shortcuts Algorithm

• New session: apple, pie
  – Apple ~ banana, orange, raspberry
    • Apple => tea, raspberry
    • Banana => tea
    • Orange => raspberry, coffee
    • Raspberry => coffee
  – Pie ~ cake, spaghetti
    • Pie => coffee
    • Cake => tea
    • Spaghetti => raspberry
  – Recommend: coffee, tea, raspberry
Experiments: Setup

- RFP 2006 dataset: randomly selected 10% of the sessions
  - More than 745,000 sessions
  - 40% of the sessions finished without any click
  - Extremely sparse dataset

- Dataset divided into training set evaluation sets

- Metrics:
  - Accuracy metrics: measure the quality of the algorithm (e.g. MAE, Good Predicted Items MAE, Good Items, MAE)
  - Classification metrics: measure the relevance of the recommended items (e.g. precision, recall, Half-Life Utility)
  - Coverage: percentage of items the algorithm is able to make a prediction for
  - Our metric: similarity score
Experiments: Results

Constrained Pearson is the best one? Not so sure ...

MAE results are not so good => mean for good and bad queries. GIM and GPIM are more realistic
Experiments: Results

- Coverage

![Bar chart showing coverage results for different methods]
Experiments: Results

No impressive results: high volume + sparsity
Experiments: Results
Conclusions

- Introduced the Search Shortcuts Discovery Problem
- Defined an evaluation framework based on query logs
- Poor results on traditional collaborative filtering and ad-hoc algorithm:
  - Low coverage => They cannot extract enough information
  - Data sparsity and rating extraction
Present work

• To reduce the sparsity => Apply preprocessing techniques:
  – Stemming
  – Stopwords

• Removal of small sessions (session threshold)

• Similarity score: great correlation with traditional precision
  – Results can be easily interpreted in terms of reduction of the session length
  – Also, can evaluate different aspects
Future work

- Improve the rating extraction
- Improve the session identification:
  - One “big” session may contain several query paths
- Study also traditional query suggestions algorithms
Thank you!

Questions?