Personalized Query Autocompletion for News Search

Lorand Dali
Blaz Fortuna
Jan Rupnik
Outline

- Introduction
- Query Ranking Model
- Dataset
- Experiments
  - Ranking evaluation
  - Dominant features
  - Saved keystrokes
- Conclusions and future work
Introduction

- The user types “mo”, the search engine suggests: “morning”, “mother”, “monastery”, “moon”, …

- Two main goals:
  - Guess the intended query after just a few keystrokes
  - Rank the query suggestions
  - Make the suggestions specific to the user
Query Ranking Model

- $x$ is the prefix of a query
- $\{F_1, F_2, \ldots, F_N\}$ set of user features
- $Q$ set of candidate queries
- $\forall q \in Q, \ p(q|F_1, F_2, \ldots, F_N)$
- $\text{score}(q) = p(q) \cdot p(q|F_1) \cdot \ldots \cdot p(q|F_N)$
Dataset

- Query logs from an online news site
- 380000 searches
- 250000 unique users (112000 registered)
  - not reg: city, country, time
  - reg: age, gender, industry, job, income
- 80000 unique queries
Experiments

- Chronologically first 300000 are the training set
- We evaluate:
  - How often the correct query is in the top 3 suggestions
  - Which features are most helpful in ranking
  - How many keystrokes the user saves
- **Baseline**: rank the queries by most frequent first
<table>
<thead>
<tr>
<th>FEATURES</th>
<th>Prefix</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>TOP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNREG.</td>
<td>1</td>
<td>11.50</td>
<td>5.08</td>
<td>2.47</td>
<td>19.06</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>23.44</td>
<td>10.12</td>
<td>5.39</td>
<td>38.96</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>43.96</td>
<td>13.53</td>
<td>7.56</td>
<td>65.05</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>56.87</td>
<td>14.59</td>
<td>5.30</td>
<td>76.78</td>
</tr>
<tr>
<td>REGISTERED</td>
<td>1</td>
<td>12.07</td>
<td>6.01</td>
<td>2.16</td>
<td>20.25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>25.56</td>
<td>9.86</td>
<td>4.90</td>
<td>40.33</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>44.44</td>
<td>13.79</td>
<td>6.54</td>
<td>64.79</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>57.71</td>
<td>13.44</td>
<td>4.86</td>
<td>76.02</td>
</tr>
<tr>
<td>BASELINE</td>
<td>1</td>
<td>9.06</td>
<td>6.19</td>
<td>2.34</td>
<td>17.60</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>23.07</td>
<td>9.81</td>
<td>5.92</td>
<td>38.83</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>42.94</td>
<td>14.55</td>
<td>6.59</td>
<td>64.05</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>57.49</td>
<td>13.48</td>
<td>6.19</td>
<td>77.17</td>
</tr>
</tbody>
</table>
Dominant Features

**Def:** \(\text{score}(q) = p(q) \cdot p(q|F_1) \cdot \cdots \cdot p(q|F_N)\)
\[
\arg\max_i [p(q|F_i)]
\]
Keystrokes Saved

- The user types the first character of $q$
  - The auto-completion algorithm is run
  - If the correct answer is in the top 3 we assume success and we saved $\text{length}(q) - 1$ keystrokes
  - Otherwise the user types one more character and we go again
  - If after 4 characters the correct query is still not in the top 3, we assume failure
- On average 5.7 keystrokes are saved (40% less typing)
Query Suggestion

- Using another approach for generating the candidate set $Q$, we can extend to query suggestions.
- **Example:** A user has in the past searched for *crossword*, we suggest related queries: *kenken, crossword puzzle, today’s crossword, puzzle, daily crosswords*, etc.
Conclusions

- We developed a simple personalized query-autocompletion model
- 75% of the time we suggest the correct query
- The user has to type 40% less
- Personalization is useful especially if the prefix is short (1 or 2 characters)
- In the future we plan to extend the work to query suggestion and include the user’s search history as a feature
Thank you!