A language independent user adaptable approach for word auto-completion

Stefan Prisca
Technical University of Cluj-Napoca,
Department of Computer Science,
Email: stefan.prisca@gmail.com

Rodica Potolea
Technical University of Cluj-Napoca,
Department of Computer Science,
Email: rodica.potolea@cs.utcluj.ro

Mihaela Dinsoreanu
Technical University of Cluj-Napoca,
Department of Computer Science,
Email: mihaela.dinsoreanu@cs.utcluj.ro
Introduction

- auto-completion is more and more frequent
  - e.g. query completion in search engines, code completion in IDEs, word/phrase completion in text editors etc.
- most offline auto-completion systems either provide completions based on
  - the document at hand (user-written) or
  - a default set of documents (default)
Objectives

Design
- Language Independency
- Phrase and Word Completion Integration
- Easy Install

Auto-completion
- User Adaptable Completions
- Fast Query Processing
Conceptual Design - Achieving design objectives

- Decouple the data models from auto-completion processors, which allows to switch between data models at runtime
  - $\Rightarrow$ Language Independence
- Separate Word Autocompletion and Phrase Autocompletion sub-systems
  - $\Rightarrow$ Word and Phrase auto-completion integration
- Public API to connect to a text editor
  - $\Rightarrow$ Easy Install
Objectives - auto-completion

- User Adaptable completions
  - provide both user-written and general auto-completion
  - prioritize user-written documents
- small query processing times!
  - research shows that for something to appear instantaneously to the human eye, it needs to appear in less than 100 ms.
Getting serious...
Formalizing word auto-completion

- All possible proposals (words) are stored in an auto-completion data model (e.g. an Inverted Index or Suffix Tree)
- Word Completion is the problem of predicting a word given a set of previous words (PW), and the first letters (FL) of that word.
- This is passed to the system in the form of a Query: $Q:\{PW: [w_1, w_2], FL: "l1l2\ldots\" \}$
- e.g.: I am go... => $Q:\{PW: [I, am], FL: "go" \}$
Word Auto-completion

- Our word completion system relies on the Inverted Index data structure

<table>
<thead>
<tr>
<th>Word</th>
<th>Posting List</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;word&gt;</td>
<td>[&lt;docId1&gt;, &lt;docId2&gt;, ...]</td>
</tr>
</tbody>
</table>

- Query processing:
  - Find all words that start with FL => matched words
  - Return all matched words that have common documents with PW
Consider the documents:

1. "I am going to the market"
2. "The market is filled with people"
3. "I hate it when people fill the market"

- Italics mark words with occurrence thresholds < 2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>1, 3</td>
<td>is</td>
</tr>
<tr>
<td>the</td>
<td>1, 2, 3</td>
<td>filled</td>
</tr>
<tr>
<td>market</td>
<td>1, 2, 3</td>
<td>with</td>
</tr>
<tr>
<td>people</td>
<td>2, 3</td>
<td>am</td>
</tr>
<tr>
<td>going</td>
<td>1</td>
<td>hate</td>
</tr>
<tr>
<td>to</td>
<td>1</td>
<td>it</td>
</tr>
<tr>
<td>when</td>
<td>3</td>
<td>fill</td>
</tr>
</tbody>
</table>

Query: \texttt{In the mar} $\Rightarrow \{ \text{PW:}[in, the], \text{FL:}”mar”\}$ $\Rightarrow$ Answer: \texttt{market}
Default and User Predictions

- Need to identify user and general documents:
  - *General Documents* = documents that are not written by the user, and that are used for initial predictions
  - *User Documents* = documents that are written by the user, after using the system for a while.
Default and User Predictions

- Use document ids to separate between user documents and general documents:
  - User Documents are incremented with a userDocMask
- Allow more user words within the index

=> An altered version of the Inverted Index, which is called User Oriented Index.

- We also store information about word positions in documents. This is used for ranking, and will be explained.
The User Oriented Index

1. Default: "I am going to the market"
2. Default: "The market is filled with people"
3. Default: "I hate it when people fill the market"
4. User written: "Today I was at the market".

- userDocMask = 100
- Occurrence Th = 2, User Occurrent Th = 0

<table>
<thead>
<tr>
<th>i</th>
<th>[1, 3, 4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>[1, 2, 3, 4]</td>
</tr>
<tr>
<td>market</td>
<td>[1, 2, 3, 4]</td>
</tr>
<tr>
<td>people</td>
<td>[2, 3]</td>
</tr>
</tbody>
</table>

The table on the right shows the inverted index and the user-oriented index.
The User Oriented Index

- Extension of the Inverted Index
- Identify user documents with a userDocMask:
  - General Document (initial prediction):
    - docId < userDocMask
  - User document (user-written):
    - docId >= userDocMask
- Store positions on which words appear in documents
  - Create word contexts
  - Compute word frequency
- Allow more words from the user in the index

<table>
<thead>
<tr>
<th>&lt;word&gt;</th>
<th>&lt;docId&gt; : [&lt;positions&gt;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>{1 : [1], 3 : [1], 101 : [2]}</td>
</tr>
<tr>
<td>the</td>
<td>{1 : [5], 2 : [1], 3 : [7], 101 : [5]}</td>
</tr>
<tr>
<td>market</td>
<td>{1 : [6], 2 : [2], 3 : [8], 101 : [6]}</td>
</tr>
<tr>
<td>people</td>
<td>{2 : [6], 3 : [5]}</td>
</tr>
<tr>
<td>today</td>
<td>{101 : [1]}</td>
</tr>
<tr>
<td>was</td>
<td>{101 : [3]}</td>
</tr>
<tr>
<td>at</td>
<td>{101 : [4]}</td>
</tr>
</tbody>
</table>

- doc1: “I am going to the market”
- doc2: “The market is filled with people”
- doc3: “I hate it when people fill the market”
- userDoc: “Today I was at the market”
Word Auto-completion Ranking

1. Frequency Score:

\[ freqScore(w, PW) = \frac{\sum_{docId} dist(w, PW, docId)}{freq(w)} \]

- \( dist() \) = the distance between all positions of a possible completion \( w \) and the previous words, in a given document
- \( freq(w) \) = the number of times word \( w \) appears in all documents
2. **User Score:**

\[
\text{score}(w, PW) = \text{freqScore}(w, PW) \times \text{userInfluence}^{uOCC}
\]

- \textit{userInfluence} = variable that we find experimentally
- \textit{uOCC} = the number of times the word \( w \) appears in user-written documents
Word Retrieval

- Fast query processing means fast word retrieval
- Index can grow up to tens of thousands of words => good word retrieval algorithm required to ensure that even with huge sizes (~ 100k words), retrieval times are below 100 ms
Word Retrieval

- The retrieval problem: retrieve a group of words, all of which start with a given prefix.
- We based our word retrieval on the binary search algorithm:
  - requires the index to always be sorted
  - reduces search times to $O(\log n)$.

=> Bidirectional Group Boundary Identification
1. **find any** word that starts with the given group of letters using binary search.
2. **create two position sentinels:**
   a. one of them decreases until the word on the current position no longer matches the letter group \(st\)
   b. the other increases until the word on the current position no longer matches the letter group \(end\)
3. **return** all words with positions in the range created by the two sentinels.
Testing - Data Sets

- Ro Small – 72,000 words, collected from blog articles and User FB messages
- En Medium – 1 million words, collected from wiki articles and User SW docs
- En Large – 7.4 million words, various web articles, with user documents about Food recipes.
Metrics

- Mean Reciprocal Rank metric for precision and recall:

\[
\text{RankPrecision} = \frac{\sum 1/\text{rank}(\text{accepted\_autocompletion})}{n(\text{predicted\_autocompletion})}
\]

\[
\text{RankRecall} = \frac{\sum 1/\text{rank}(\text{accepted\_autocompletion})}{n(\text{queries})}
\]
## Results

**Simple Index vs User Oriented Index**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>Runtime (ms)</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB Messages (Ro Small)</td>
<td>User Oriented</td>
<td>80%</td>
<td>78%</td>
<td>4</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>Simple Index</td>
<td>71%</td>
<td>68%</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Sw Products (En Medium)</td>
<td>User Oriented</td>
<td>89%</td>
<td>87%</td>
<td>1</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>Simple Index</td>
<td>71%</td>
<td>61%</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Food Recipes (En Large)</td>
<td>User Oriented</td>
<td>84%</td>
<td>82%</td>
<td>6</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Simple Index</td>
<td>76%</td>
<td>66%</td>
<td>6</td>
<td>40</td>
</tr>
</tbody>
</table>
User Oriented Index - Learning

- the User Oriented system has a bigger learning step.
- around 3800 words both systems learn relevant content:
  - User Oriented increases its precision with 20%
  - Simple Index increases its precision with 13%.
- => learning capabilities increased with 53% over the Simple Index
Bidirectional Group Boundary Identification - Runtime comparison

- BGBI keeps low runtimes even with large indexes
- => on average, it reduced the word retrieval runtime with 80%
Conclusion

- Presented a language independent system design for word and phrase auto-completion
- Introduced a data model that increases the performance of word auto-completion systems by learning from the user (User-Oriented Index)
- Developed a fast binary search algorithm that decreases word retrieval time by 80% (Bidirectional Group Boundary Identification)