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

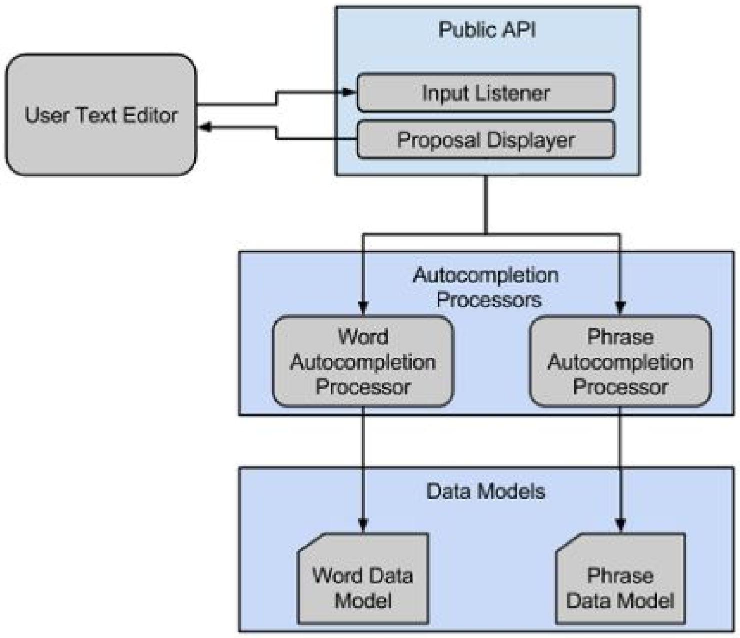
Design

- 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
  - => *Language Independency*
- Separate Word Autocompletion and Phrase Autocompletion sub-systems
  - => *Word and Phrase auto-completion integration*
- Public API to connect to a text editor
  - => *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:[w1, w2], FL:”l1l2… ”}*
- 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

Word	Posting List
<word>	[<docId1>, <docId2>, ...]

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

The Inverted Index

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

i	[1, 3]	*is*	*[2]*
the	[1, 2, 3]	*filled*	*[2]*
market	[1, 2, 3]	*with*	*[2]*
people	[2, 3]	*am*	*[1]*
going	*[1]*	*hate*	*[3]*
to	*[1]*	*it*	*[3]*
when	*[3]*	*fill*	*[3]*

Query:***In the mar*** ⟹ *{PW:[in, the], FL:”mar”}* ⟹ *Answer:* ***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 *userDocM*ask
- 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

i	[1, 3, 4]
the	[1, 2, 3, 4]
market	[1, 2, 3, 4]
people	[2, 3]

Inverted Index

→

I	{1 : [1], 3 : [1], 101:[2]}
the	{1 : [5], 2 : [1], 3 : [7], 101 : [5]}
market	{1 : [6], 2 : [2], 3 : [8], 101 : [6]}
people	{2 : [6], 3 : [5]}
today	{101 : [1]}
was	{101 : [3]}
at	{101 : [4]}

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

- 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>	{<docId> : [<postitions>]}

i	{1 : [1], 3 : [1], 101:[2]}
the	{1 : [5], 2 : [1], 3 : [7], 101 : [5]}
market	{1 : [6], 2 : [2], 3 : [8], 101 : [6]}
people	{2 : [6], 3 : [5]}
today	{101 : [1]}
was	{101 : [3]}
at	{101 : [4]}

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

Word Auto-completion Ranking

2. **User Score**:

$$score(w, PW) = freqScore(w, PW) * userInfluence^{uOcc}$$

- *userInfluence* = variable that we find experimentally
- uOCC = the number of times the word w appears in user-written documents

Word Retrieval

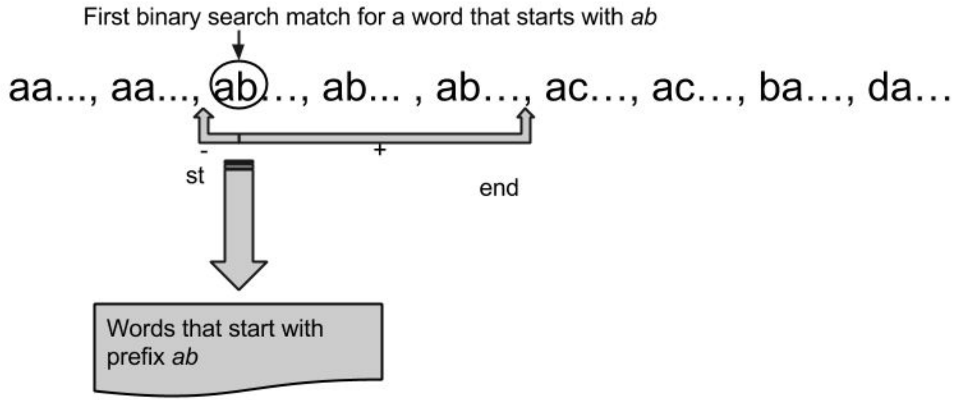
- Fast query processing means fast word retrieval
- Index can grow up to tens of thousands of words => good word retrieval algorithm required to ensure that even with huge sizes (~ 100k words), retrieval times are below 100 ms

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*

Word Auto-completion - Word Retrieval: *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:

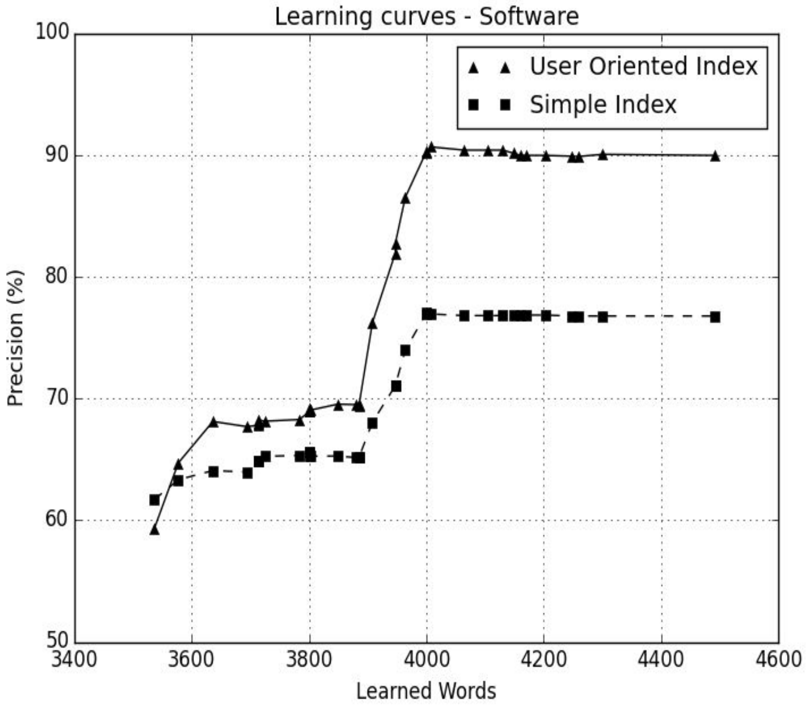
$$RankPrecision = \frac{\sum 1/rank(accepted\_autocompletion)}{n(predicted\_autocompletion)}$$

$$RankRecall = \frac{\sum 1/rank(accepted\_autocompletion)}{n(queries)}$$

Results
Simple Index vs User Oriented Index

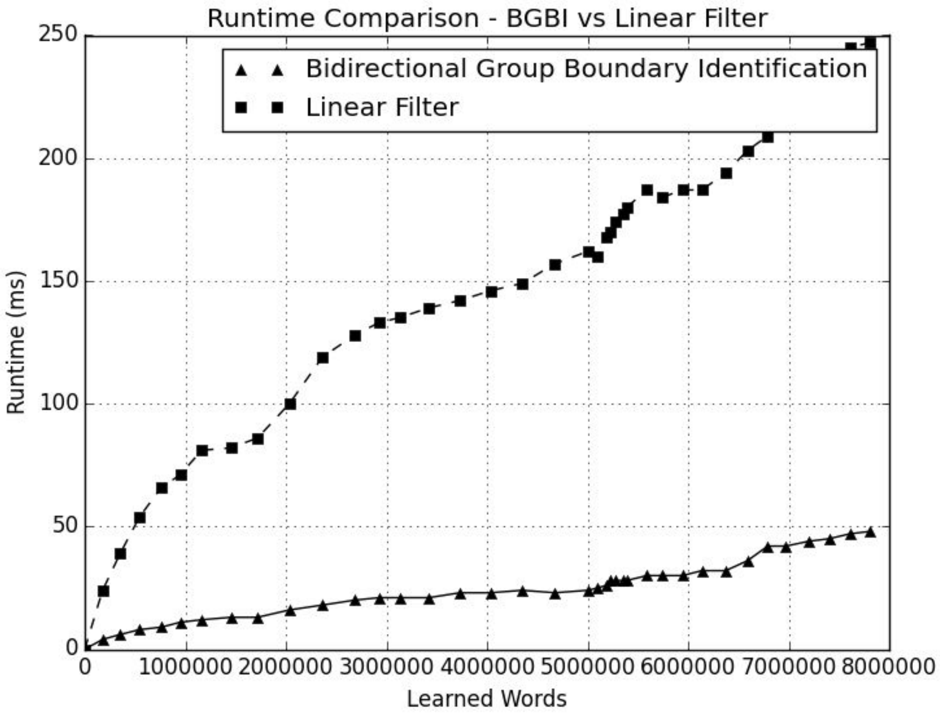
Data Set	Model	Precision	Recall	Runtime (ms)	Size (MB)
FB Messages (Ro Small)	User Oriented	80%	78%	4	4.3
	Simple Index	71%	68%	4	1
Sw Products (En Medium)	User Oriented	89%	87%	1	6.5
	Simple Index	71%	61%	1	1
Food Recipes (En Large)	User Oriented	84%	82%	6	48
	Simple Index	76%	66%	6	40

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)*