Query Auto-Completion using an Abstract Language Model

BACHELOR THESIS BY NATALIE PRANGE, 02.11.2016

Introduction to query auto-completion

- Query auto-completion (QAC): suggesting completions for a query prefix entered by a user

- Objective:
  - Reduce the user‘s effort to enter a query
  - Prevent spelling mistakes
  - Assist in formulating a query

  → A QAC-algorithm must suggest the desired query after a minimal amount of keystrokes at a high rank

A common solution

- Suggest the most popular queries from a query log that match the given prefix
- Problems with this approach:
  - Recent and large enough query logs are hard to get
  - Queries which are asked for the first time are not suggested

A language-model-based solution

- Focus in this work is on whole questions

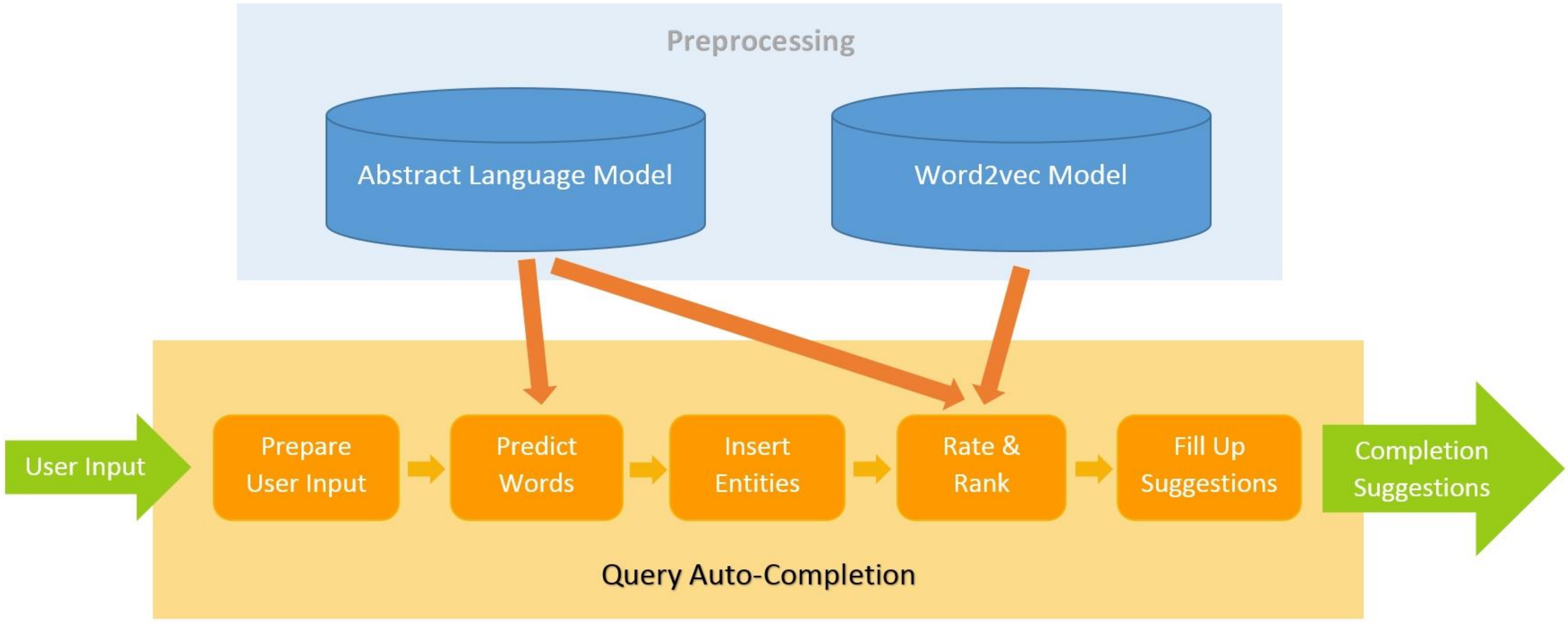
  → possible solution: use a language model

- Language model = probability distribution learned over sequences of words
- Can be used to predict the word most likely to follow a given sequence
- Typical problem: data sparsity

This approach

- Use an abstract language model: specific entities are replaced by abstract types
  - E.g.: *„Who played Gandalf in The Lord of the Rings ?“*

    → *„Who played [fictional character] in [film] ?“*
- When the language model predicts a type, entities are inserted again
- A prominence score and word vector similarity are used to rank suggestions



Basic pipeline of the Auto-Completion algorithm.

Building the abstract language model

- Choosing a type for each entity:
  - Out of a list of types of an entity, choose the most general but still meaningful type
  - E.g.: *Albert Einstein: [Person, Astronomer, Diet Follower, Topic, …]* → *Person*
  - Choose a type according to a hand-picked list of preferred types

Building the abstract language model

- The training set consists of questions in which recognized entities are replaced by their type
- An n-gram language model is learned on these questions
- N-gram model:
  - Estimate the probability of a word given it‘s (n-1) predecessors:
  - $P(w_m|w_1, \dots, w_{m-1}) \approx P(w_m|w_{m-(n-1)}, \dots, w_{m-1}) = \frac{count(w_{m-(n-1)},\dots,w_{m-1},w_m)}{count(w_{m-(n-1)},\dots,w_{m-1})}$

Building the Word2vec model

- Word2vec uses a neural network to learn vector representations of words
- The more common context two words share, the higher the cosine similarity of their word vectors

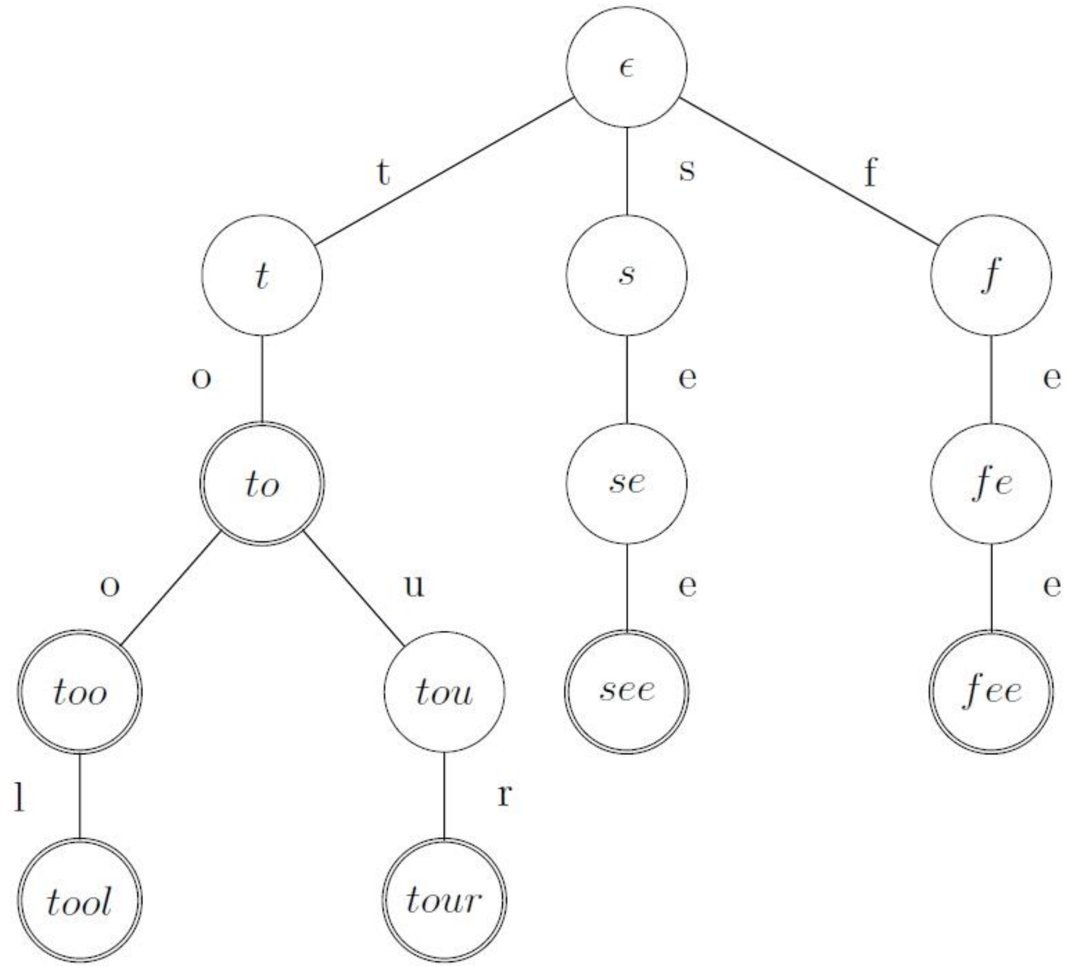
  → can be used to compute semantic similarity between words
- E.g.: *vector(Berlin) – vector(Germany) + vector(France) ≈ vector(Paris)*

Predicting possible next words

- Normally:
  - last (n-1) complete words = n-gram context
  - last incomplete word = prefix of the next word
- Here: a predicted type can correspond to multiple words typed by the user
  - E.g.: *„Who played [Fictional Character|Iron Man] in the first A"*
  - → Which words are part of an entity name and which are normal words?
- Get predictions for all possible prefixes and their corresponding n-gram context

Inserting entities for types

- Insert entities for every type predicted by the n-gram model
- Entities need to match the given type and match the given prefix
- Prefix trees are used for retrieval of entities



Prefix tree, built from the words [to, too, tool, tour, see, fee]

Rating and ranking

- 1st scenario: the question prefix does not contain any entity
  - Use a prominence score to rate entities
  - Normalize score
  - $s_{final} = p_{n-gram} * (s_{norm})^{0.3}$
- 2nd scenario: the question prefix contains at least one entity
  - Compute word vector similarity between the contained, and the suggested entity
  - Fill in the word vector similarity for s_{norm}
- Normal words are assigned a fixed score in both approaches

Filling up the completion suggestions

- Use words that were not predicted by the n-gram model
- Use the prominence score and word count for rating the fill-up words
- Always append completely typed entities to the completion suggestions

Evaluation: Metrics

- User Interaction:
  - $\frac{\text{(total keystrokes + total selections)}}{\text{(total number of characters in question)}}$
- Mean Reciprocal Rank (MRR):
  - q_c: matching completion suggestion, S: completion suggestions
  - $RR(q_c, S) = \frac{1}{rank(q_c, S)}$
  - RR is computed after typing the first letter of a word
  - MRR is the mean of the RR‘s of every word / entity name in every question
- Percentage of unidentified entities

Evaluation: Tested algorithm versions

- Baseline:
  - Without filling up completion suggestions
  - Without appending complete words
- 2nd Version: Without appending complete words
- 3rd Version: Only prominence score for rating (no word vectors)
- 4th Version: Complete algorithm as described

Evaluation: Results

All questions				
Algorithm Version	MRR	User Interaction	Unid. Entities	Time
Baseline	0.376	0.64	38.9%	0.027 secs
w/o complete entities	0.469	0.49	11.1%	0.047 secs
w/o Word2vec model	0.449	0.49	6.3%	0.040 secs
Complete algorithm	0.457	0.49	6.3%	0.047 secs

Evaluation: Results

Questions containing one entity				
Algorithm Version	MRR	User Interaction	Unid. Entities	Time
Baseline	0.373	0.64	33.2%	0.028 secs
No complete entities	0.469	0.49	10.2%	0.048 secs
w/o Word2vec model	0.449	0.50	6.2%	0.041 secs
Complete algorithm	0.457	0.50	6.2%	0.047 secs

Questions containing two or more entities				
Algorithm Version	MRR	User Interaction	Unid. Entities	Time
Baseline	0.385	0.66	50.4%	0.025 secs
No complete entities	0.465	0.49	15.7%	0.046 secs
w/o Word2vec model	0.444	0.47	6.7%	0.037 secs
Complete algorithm	0.452	0.48	6.8%	0.046 secs

Completion suggestions using the Word2vec model:

who played [fictional_character|Gollum] in th

- who played [fictional_character|Gollum] in the
- who played [fictional_character|Gollum] in [film|The Lord of the Rings: The Fellowship of the Ring]
- who played [fictional_character|Gollum] in [film|The Lord of the Rings: The Return of the King]
- who played [fictional_character|Gollum] in [film|The Doctor]
- who played [fictional_character|Gollum] in [netflix_title|The Beast]

Completion suggestions using only an entity prominence score:

who played [fictional_character|Gollum] in th

- who played [fictional_character|Gollum] in the
- who played [fictional_character|Gollum] in [film|The Hunger Games (Science Fiction Film)]
- who played [fictional_character|Gollum] in [film|The Corporation]
- who played [fictional_character|Gollum] in [film|The Queen]
- who played [fictional_character|Gollum] in [tv_program|The Today Show]

Future work

- Integrate proper entity recognition
  - E.g.: *USA* → *United States of America*
- Robustness against spelling mistakes
- Multiple-word suggestions