



# Personalized Query Autocompletion for News Search

Lorand Dali  
Blaz Fortuna  
Jan Rupnik

# Outline

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

# Introduction

- o The user types “**mo**”, the search engine suggests: “**morning**”, “**mother**”, “**monastery**”, “**moon**”, ...
- o Two main goals:
  - o Guess the intended query after just a few keystrokes
  - o Rank the query suggestions
- o Make the suggestions specific to the user

# Query Ranking Model

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

# Dataset

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

# Experiments

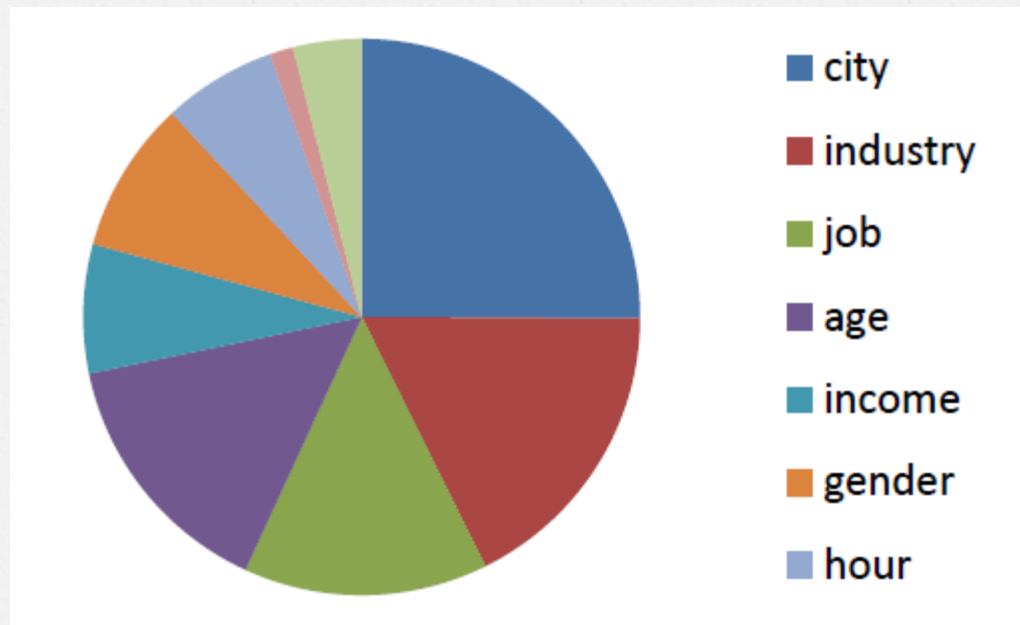
- o Chronologically first 300000 are the training set
- o We evaluate:
  - o How often the correct query is in the top 3 suggestions
  - o Which features are most helpful in ranking
  - o How many keystrokes the user saves
- o Baseline: rank the queries by most frequent first

**RANK OF SUGGESTION (%)**

<b>FEATURES</b>	<b>Prefix</b>	<b>R1</b>	<b>R2</b>	<b>R3</b>	<b>TOP3</b>
<b>UNREG.</b>	<b>1</b>	11.50	5.08	2.47	19.06
	<b>2</b>	23.44	10.12	5.39	38.96
	<b>3</b>	43.96	13.53	7.56	65.05
	<b>4</b>	56.87	14.59	5.30	76.78
<b>REGISTERED</b>	<b>1</b>	12.07	6.01	2.16	20.25
	<b>2</b>	25.56	9.86	4.90	40.33
	<b>3</b>	44.44	13.79	6.54	64.79
	<b>4</b>	57.71	13.44	4.86	76.02
<b>BASELINE</b>	<b>1</b>	9.06	6.19	2.34	17.60
	<b>2</b>	23.07	9.81	5.92	38.83
	<b>3</b>	42.94	14.55	6.59	64.05
	<b>4</b>	57.49	13.48	6.19	77.17

# Dominant Features

- Def:  $score(q) = p(q) \cdot p(q|F_1) \cdot \dots \cdot p(q|F_N)$   
 $argmax_i [p(q|F_i)]$





# Keystrokes Saved

- o The user types the first character of  $q$ 
  - o The auto-completion algorithm is run
  - o If the correct answer is in the top 3 we assume **success** and we saved  $length(q) - 1$  keystrokes
  - o Otherwise the user types one more character and we go again
- o If after 4 characters the correct query is still not in the top 3, we assume **failure**
- o 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

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



Thank you!

