

Context-Aware Query Suggestion by Mining Click-Through and Session Data

Presenters Tianzi Hou & Chen Wang

Presentation @ Data Mining and Exploration Course
School of Informatics, University of Edinburgh

OVERVIEW

Introduction

Challenges

Approaches

Experiments

Conclusion

Our thoughts

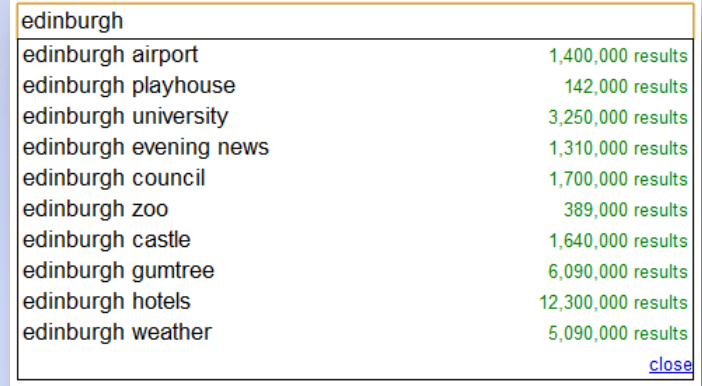
INTRODUCTION

About this paper

- Authors: Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen and Hang Li
- Publication: ACM KDD2008
- Citation: 20 (according to Google Scholar)

Query suggestion

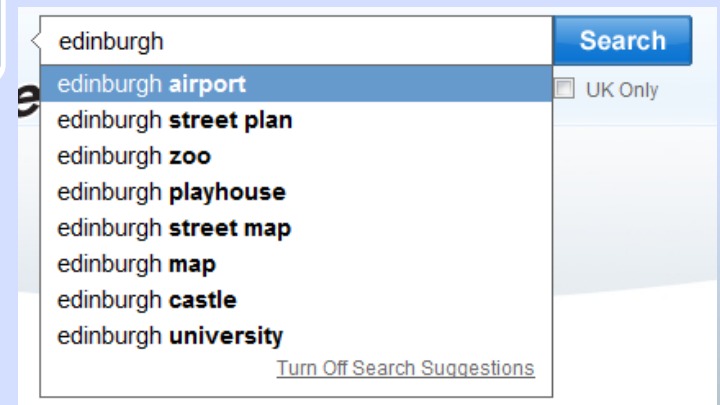
- A way to improve usability (Google, Ask.com, etc)
- Short query, ambiguous words, different result
- Similar queries in log, pairs in same query sessions
- Context-aware?



Query	Results
edinburgh	
edinburgh airport	1,400,000 results
edinburgh playhouse	142,000 results
edinburgh university	3,250,000 results
edinburgh evening news	1,310,000 results
edinburgh council	1,700,000 results
edinburgh zoo	389,000 results
edinburgh castle	1,640,000 results
edinburgh gumtree	6,090,000 results
edinburgh hotels	12,300,000 results
edinburgh weather	5,090,000 results

[close](#)

Google's query suggestion



Ask.com's query suggestion

CHALLENGES(1)

Related Work

Traditional approaches

- Session-based approaches
- Cluster-based approaches

Problems

- High computational cost
- Cannot scale up to large data
- “Curse of dimensionality”
- Cannot support the dynamic update

CHALLENGES(2)

Critical Issues

Questions

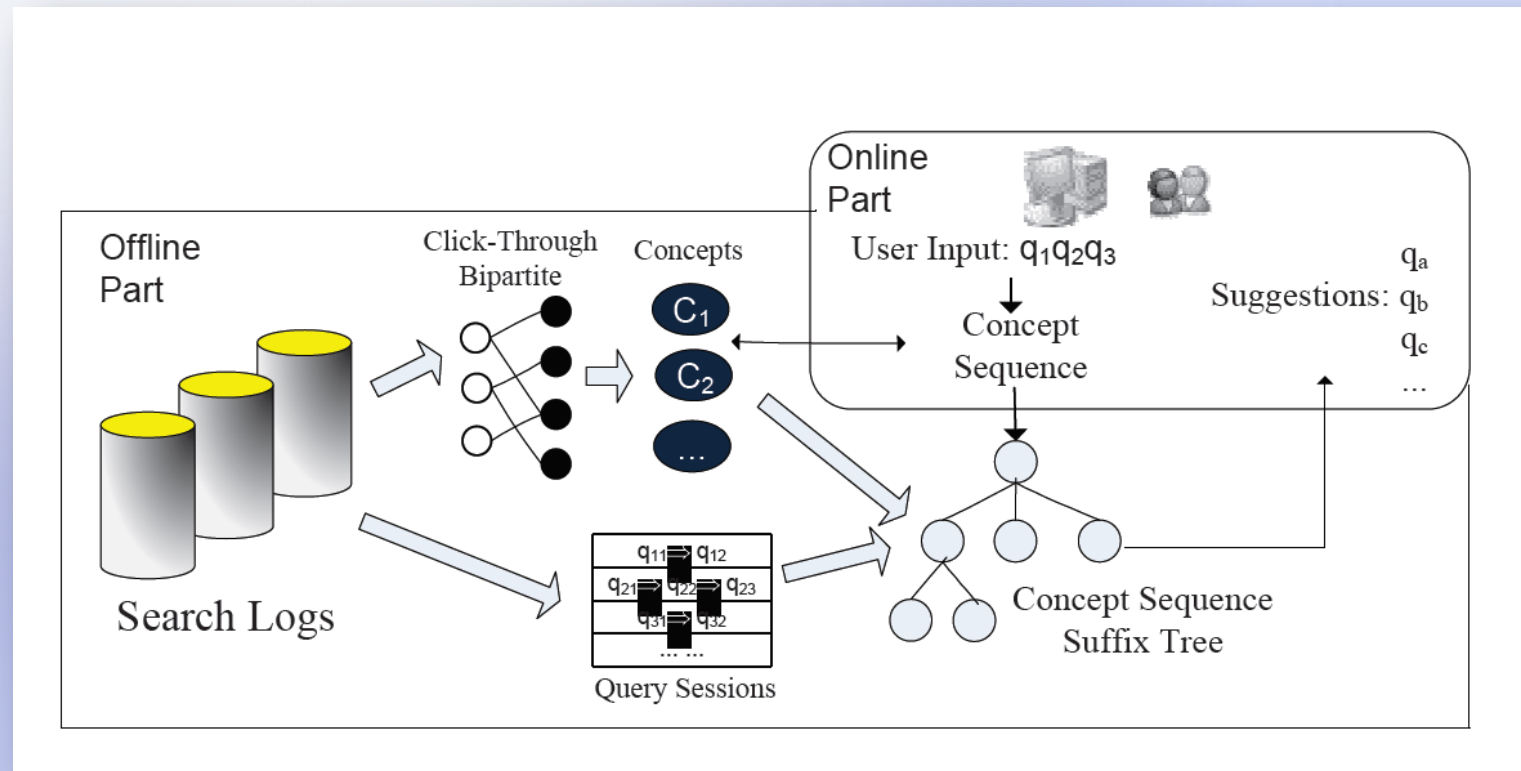
- How should we model and capture contexts well?
- How to find the queries that many users often ask in a particular context?

Critical issues

- Cluster the queries
- Reduce the computational cost
- Increase the coverage and accuracy

APPROACHES(1)

Framework



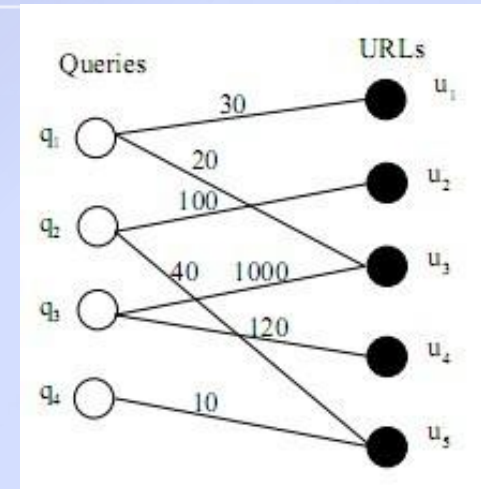
The framework of the context-aware approach

APPROACHES(2)

Mining Query Concepts

Query Representing

- Click-Through bipartite
- A query is represented as a vector of URLs
- In a vector $q[i]$,
 $q[i][j] = \text{normalized}(w[i][j])$
- where $w[i][j]$ is the number of times when URL j is a click of query i



Distance between Queries

- Euclidean distance
 $\text{distance}(q[i], q[j]) = \sqrt{\sum (q[i][k] - q[j][k])^2}$

APPROACHES(3)

Mining Query Concepts

Clustering Method

- For a query q in the query list
 - Find the closest cluster C to q obtained so far
 - If the distance between q and any query in C is less than D_{max} , assign q into C
 - Else create a cluster only contain q

Find the closest cluster fast

- Click-through bipartite is sparse
 - a query has an average number of 8.2 clicked URLs
 - an URL is a click of only 1.8 queries
 - For a query q , the number of queries which share at least one URL with q is only $8.2 * (1.8 - 1) = 6.56$

APPROACHES(4)

Conducting Query Suggestion

Query Session

- A list of query events by one individual user
- Two consecutive events are segmented into two sessions if the time interval between them exceeds 30 minutes
- Queries in a same session are often related

Query Relation	Session
Spelling correction	MSN messnger ⇒ MSN messenger
Peer queries	SMTP ⇒ POP3
Acronym	BAMC ⇒ Brooke Army Medical Center
Generalization	Washington mutual home loans ⇒ home loans
Specialization	Nokia N73 ⇒ Nokia N73 themes ⇒ free themes Nokia N73

APPROACHES(5)

Constructing Query Concepts

Concept Sequence

- Map each query sequence $qs=q[1]...q[l]$ into a sequence of concepts $cs=c[1]...c[l]$
- For a frequent concept sequence cs , if a user is searching $cs'=c[1]...c[l-1]$, the queries in $c[l]$ can be our suggestions

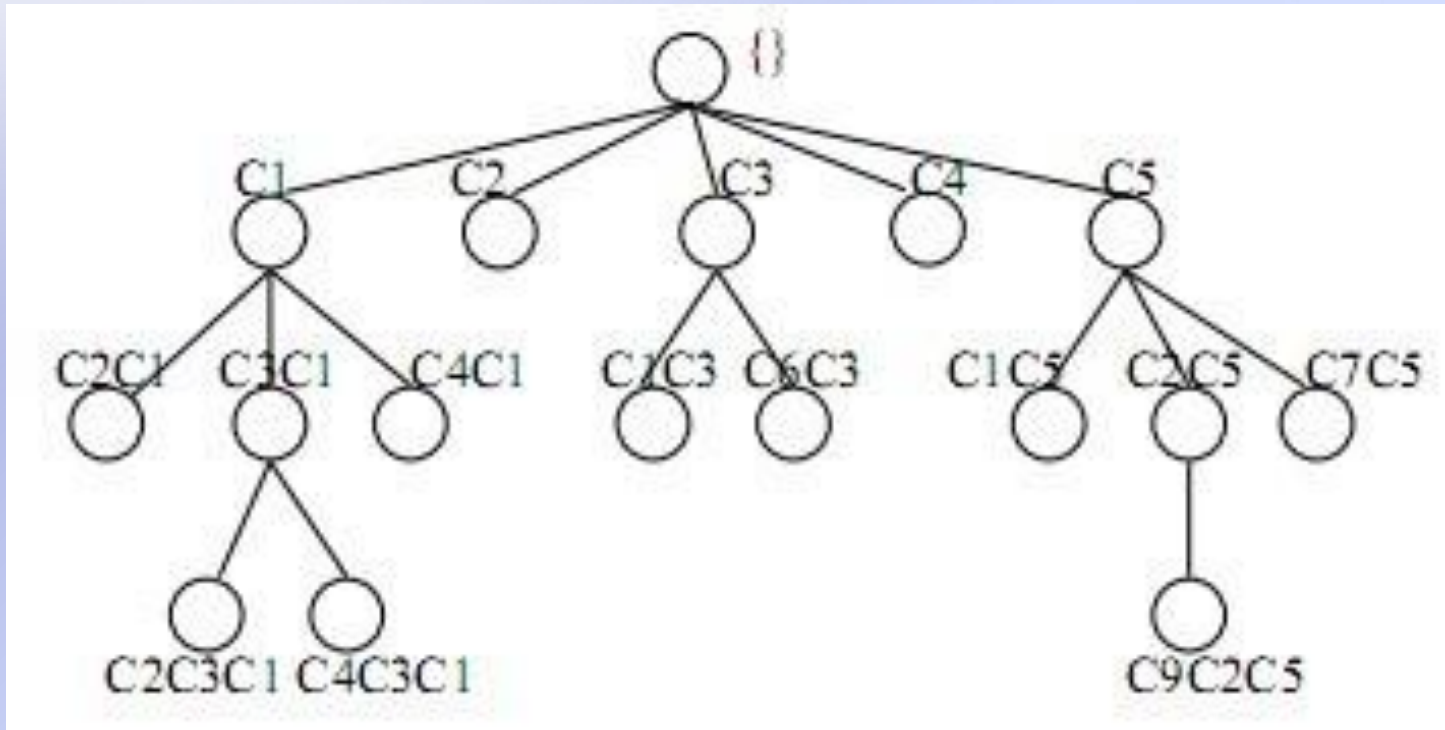
Frequent Concept Sequence

- Number of concept sequence is far less than the number of possible combinations of concepts
- We only concern about the subsequence of the concept sequence
- The average number of concepts in a session is usually small

APPROACHES(6)

Conducting Query Suggestion

Concept Sequence Suffix Tree



APPROACHES(7)

Constructing Query Concepts

Building the Tree

- For a frequent concept sequence $cs=c[1]...c[l]$
 - Find the node n corresponding to $cs'=c[1]...c[l-1]$
 - If cn does not exist, create a new node for cs' recursively
 - Update the node with candidate suggestions in $c[l]$

Online Query suggestion

- Mapping the query sequence into concept sequence cs
- Find the node matches cs most
- Return the candidate suggestions of the node

EXPERIMENTS(1)

Brief Introduction

Scale

- 1.8 billion search queries
- 2.6 billion clicks
- 840 million query sessions (151 million unique queries, 114 million unique URLs)

Scope

- English
- US market

Contents

- Clustering the Click-Through Bipartite
- Building the Concept Sequence Suffix Tree
- Evaluation of Query Suggestions
- Robustness and Scalability of Algorithms

EXPERIMENTS(2)

Experimental Results

Cluster click-through bipartite

- The click-through bipartite is sparse
- 200 thousand clusters cover 700 thousand queries
- Remaining queries form singleton clusters

Build concept sequence suffix tree

- The session length is between 1 and 10
- Prune the nodes containing more than 4 concepts

EXPERIMENTS(3)

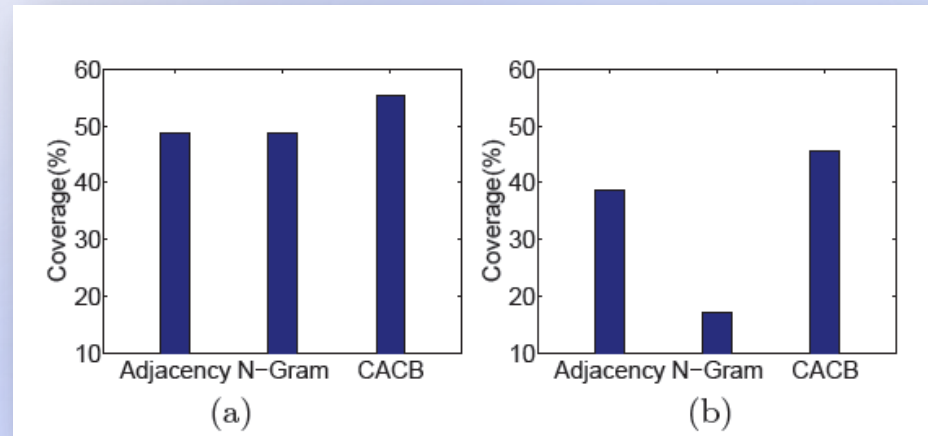
Evaluation of Query Suggestions

Methods

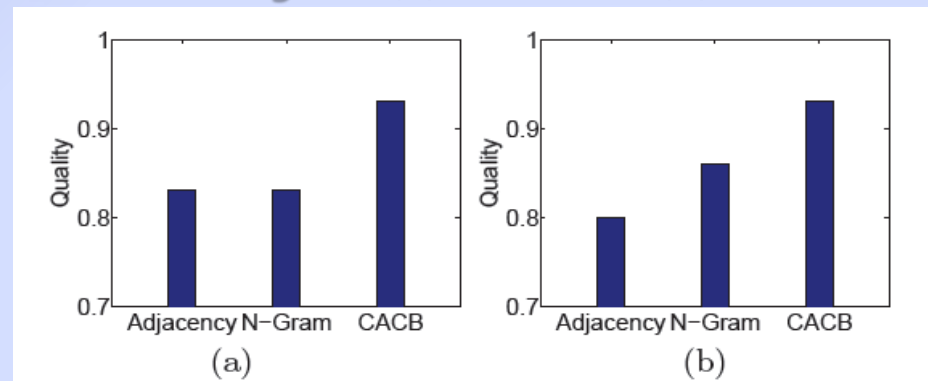
- Adjacency
- N-Gram
- CACB

Tests

- Test-0
- Test-1



The coverage on (a) Test-0 and (b) Test-1



The quality on (a) Test-0 and (b) Test-1

EXPERIMENTS(4)

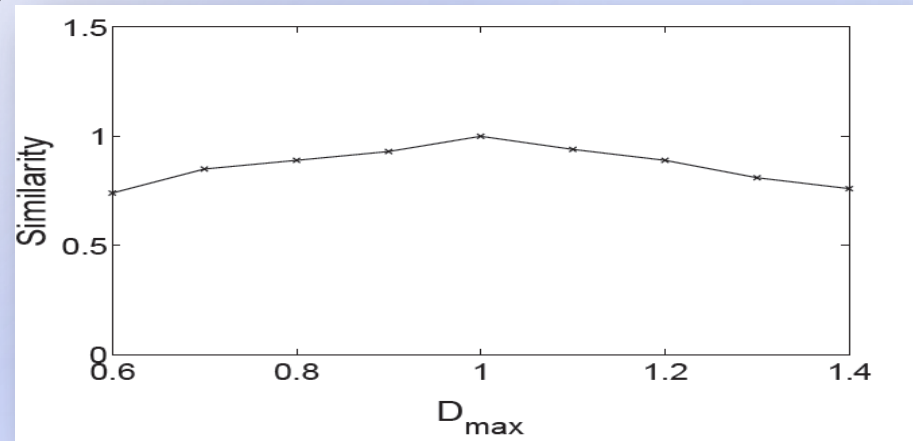
Robustness and Scalability

Robustness

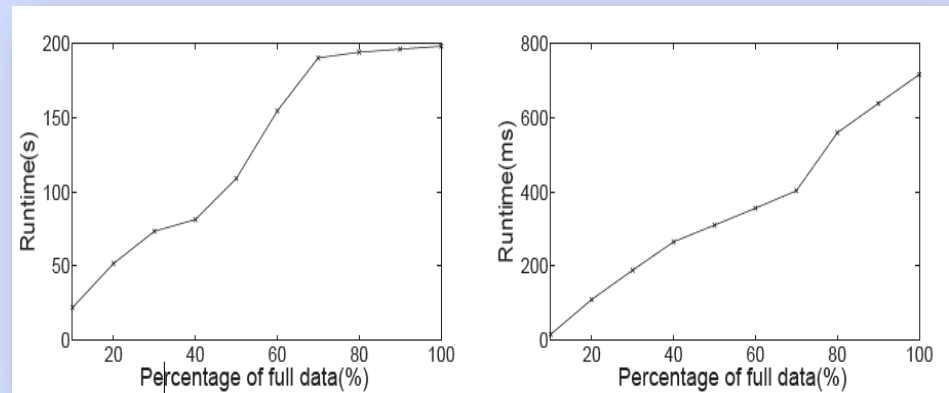
- Compare with $\frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$
- Do not change more

Scalability

- Almost linear to the input size
- 200s for clustering queries (100% data)
- < 1s for building tree (100% data)



(a) The robustness of the clustering algorithm



(b) Clustering queries (c) Building the suffix tree

CONCLUSION

- Challenges
 - Large amount of data
 - High dimension of data
 - Dynamic update
- Approach
 - Clustering
 - Concept sequence suffix tree
- Result
 - Outperform two baselines in both coverage and quality

OUR THOUGHTS

- Limitations of the context-aware query suggestion

THANK YOU!