

# Context Search the case of Wikipedia

Debora Donato

Yahoo! Research – Barcelona, Spain

Seminars of Computer Networks

# Outline

- 1 Introduction
- 2 Approach
- 3 Experiments

- 1 Introduction
- 2 Approach
- 3 Experiments

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

# Motivation

## Keyword-based Search

$$q = \langle t_1, t_2, \dots, t_n \rangle$$

Goal: finding a ranked list of documents from a given collection

Problems: ambiguity, incomplete information.

## Context-Sensitive Search

$$\langle C, q \rangle = \langle C, t_1, t_2, \dots, t_n \rangle$$

Goal: increasing relevance.

## Example from [Botev and Shanmugasundaram, 2005]

Large XML Library ranging from Shakespeare's plays to scientific papers

- Context (user's interest): Shakespeare's plays
- $q = \langle \text{speech AND process} \rangle$
- XPath query: `//Play[author='William Shakespeare']`

Goal: limiting the scope of her search.

## Example from [Botev and Shanmugasundaram, 2005]

Large XML Library ranging from Shakespeare's plays to scientific papers

- Context (user's interest): Shakespeare's plays
- $q = \langle \text{speech AND process} \rangle$
- XPath query: `//Play[author='William Shakespeare']`

Goal: limiting the scope of her search.

## Example from [Botev and Shanmugasundaram, 2005]

Large XML Library ranging from Shakespeare's plays to scientific papers

- Context (user's interest): Shakespeare's plays
- $q = \langle \text{speech AND process} \rangle$
- XPath query: `//Play[author='William Shakespeare']`

Goal: limiting the scope of her search.

## Example from [Botev and Shanmugasundaram, 2005]

Large XML Library ranging from Shakespeare's plays to scientific papers

- Context (user's interest): Shakespeare's plays
- $q = \langle \text{speech AND process} \rangle$
- XPath query: `//Play[author='William Shakespeare']`

Goal: limiting the scope of her search.

## Example: Search while browsing

User reading a particular page  $\rightarrow$  need of new information

- Context: article about Euler
- $q = \langle \text{problem of seven bridges} \rangle$

## Example: Search while browsing

User reading a particular page  $\rightarrow$  need of new information

- Context: article about Euler
- $q = \langle \text{problem of seven bridges} \rangle$

## Example: Search while browsing

User reading a particular page  $\rightarrow$  need of new information

- Context: article about Euler
- $q = \langle \text{problem of seven bridges} \rangle$

## Example: Link finding

Wikipedia editor revising document —  $\rightarrow$  quick finding of a good link

- Context: article editor is editing
- $q = \langle \text{phrase to be linked} \rangle$

## Example: Link finding

Wikipedia editor revising document —  $\rightarrow$  quick finding of a good link

- Context: article editor is editing
- $q = \langle \text{phrase to be linked} \rangle$

## Example: Link finding

Wikipedia editor revising document —  $\rightarrow$  quick finding of a good link

- Context: article editor is editing
- $q = \langle \text{phrase to be linked} \rangle$

# Challenges

## 1st Challenge

Efficiently find search results in the search context without having to touch irrelevant content

## 2nd Challenge

Effectively rank keyword search query evaluated in a search context

- 1 Introduction
- 2 Approach**
- 3 Experiments

# Framework

- $G = (V, E)$ 
  - $G$  nodes with text information;
  - $E$  hyperlinks (?)
- query:  $\langle q, p \rangle$ 
  - $q = \langle t_1, t_2, \dots, t_n \rangle$
  - $p \in V$
- task: finding nodes  $\{x\} \in V$  relevant to the query pair  $\langle q, p \rangle$

# Framework

- $G = (V, E)$ 
  - $G$  nodes with text information;
  - $E$  hyperlinks (?)
- query:  $\langle q, p \rangle$ 
  - $q = \langle t_1, t_2, \dots, t_n \rangle$
  - $p \in V$
- task: finding nodes  $\{x\} \in V$  relevant to the query pair  $\langle q, p \rangle$

# Framework

- $G = (V, E)$ 
  - $G$  nodes with text information;
  - $E$  hyperlinks (?)
- query:  $\langle q, p \rangle$ 
  - $q = \langle t_1, t_2, \dots, t_n \rangle$
  - $p \in V$
- task: finding nodes  $\{x\} \in V$  relevant to the query pair  $\langle q, p \rangle$

# Framework

- $G = (V, E)$ 
  - $G$  nodes with text information;
  - $E$  hyperlinks (?)
- query:  $\langle q, p \rangle$ 
  - $q = \langle t_1, t_2, \dots, t_n \rangle$
  - $p \in V$
- task: finding nodes  $\{x\} \in V$  relevant to the query pair  $\langle q, p \rangle$

# Graph Theoretical Approach

- Inspired by Personalized PageRank [Haveliwala, 2002]
- Given  $\langle q, p \rangle$ , the main idea is performing PPR on  $G$ :
  - $\alpha \rightarrow$  random walk;
  - $1 - \alpha \rightarrow$  return back to page  $p$ .

# Efficiency Issues

## Scalability

**time:** unfeasible the computation at query time

**space:** one vector for each node.

## Solutions

- Page-specific PageRank with jumps to a smaller number of landmarks.
- Clustering of the graph and computation of one single cluster-specific PageRank for all pages in the cluster.

# Other features

## Approach

- **Link-based:**
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- **Content-based:**
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Other features

## Approach

- Link-based:
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- Content-based:
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Other features

## Approach

- Link-based:
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- Content-based:
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Other features

## Approach

- Link-based:
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- Content-based:
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Other features

## Approach

- Link-based:
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- Content-based:
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Other features

## Approach

- Link-based:
  - Page-specific PageRank with teleport vector to the source page
  - Distance from the source to the target page
- Content-based:
  - Traditional IR and Web search: TF/IDF, BM25, etc.
  - Textual similarity between the source and the target page

# Ranking Algorithm

- Given  $\langle q, p \rangle$  and a candidate set  $C \subseteq V$ , compute for each  $u \in C$  the feature vector  $\mathbf{x}_{\langle q, p \rangle}^u$ .
- Final score of  $u$  is given by a ranking function  $\mathbf{w}^T \mathbf{x}_{\langle q, p \rangle}^u$ , where  $\mathbf{w}$  is a vector of weights.
- RankSVM [Joachims, 2002] to compute an optimal  $\mathbf{w}$ .

# Candidate Set

Classical text-based IR approaches do not take the structure into account

- documents ranked with respect their global relevance,
- ambiguous terms

Solution: Pruning unwanted documents from the candidate set

# Candidate Set

Classical text-based IR approaches do not take the structure into account

- documents ranked with respect their global relevance,
- ambiguous terms

Solution: Pruning unwanted documents from the candidate set

# Procedure for Computing the Candidate Set

## Observation

Most of the “correct” target nodes to many queries are within short distance from the query node. The underlying intuition is that documents corresponding to nodes close to  $p$ , in terms of graph distance, are more likely to be better answers to an ambiguous query.

## Procedure

- 1 Use a pre-computed inverted index to get all documents that contain all query terms.
- 2 Run a breadth-first search (BFS) starting from the query node following the forward links up to depth  $h$ .
- 3 Let  $C$  contain all nodes that are in the intersection of the documents containing  $q$  and nodes visited by BFS.

# Ranking Features

## Content-based features

BM25:

$$\sum_{i=1}^m \text{IDF}(q_i) \cdot \frac{f(q_i, d)(k_1 + 1)}{f(q_i, d) + k_1(1 - b + b|d|/\bar{d})},$$

where  $\text{IDF}(q_i)$  is the *inverse document frequency* of the query term  $q_i$ , typically defined as

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}.$$

**Textual similarity:** Let  $T(d_v)$  be the set of terms contained in document  $d_v$ . The textual similarity is defined as  $J(T(d_u), T(d_v))$  (Jaccard coefficient).

# Ranking Features

## Link-based features

**Predecessor similarity** : Let  $P(v) = \{u \in V | (u, v) \in E\}$   
the predecessor similarity (PRED) is  $J(P(u), P(v))$ .

**Successor similarity** : Let  $S(v) = \{u \in V | (v, u) \in E\}$   
the successor similarity (SUCC) is  $J(S(u), S(v))$ .

**Spectral Distance** : an embedding  $\phi : V \rightarrow \mathbb{R}^m$  of the graph to a low dimensional Euclidean space and consider the distances of the nodes in this space.

- 1 Laplacian matrix:  $\mathcal{L}_G = D - A$ , where  $D$  is a diagonal matrix having  $A_{ii} = d_i$ .
- 2 projection  $\phi : V \rightarrow \mathbb{R}^m$  ( $(m + 1)$  eigenvectors of  $\mathcal{L}_G$ )
- 3 The spectral distance (SD) is the Euclidean distance between  $\phi(u)$  and  $\phi(v)$ .

# Ranking Function

Given the query  $\langle q, p \rangle$ , we define the score of a node  $u$  as

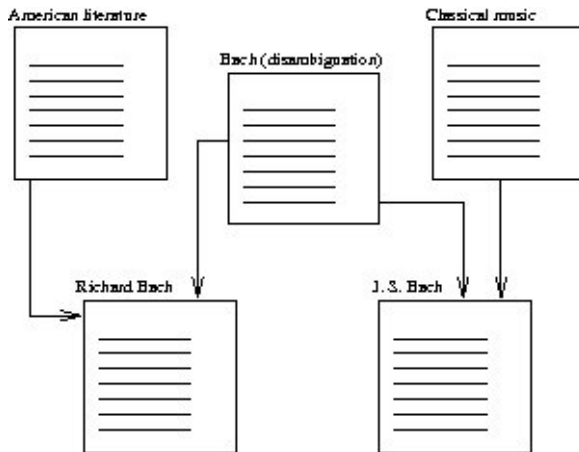
$$\text{score}(u) = \mathbf{w}^T \mathbf{x}_{\langle q, p \rangle}^u.$$

We use RankSVM [Joachims, 2002] is a supervised method that solves this optimization problem.

Training set: ambiguous queries in Wikipedia

- 1 Introduction
- 2 Approach
- 3 Experiments**

# Evaluation



**Figure:** An illustration of the use of Wikipedia disambiguation pages for finding ambiguous query terms (“bach”) for different contexts (“American literature” or “Classical music”).

# Individual features without BFS

	success rate @ k			mean	median
	k=1	k=5	k=10		
BM25	0	12	25	285	38
TEXT	0	21	28	3435	121
SUCC	0	21	30	1634	56
PRED	0	23	34	1308	63
SD	1	12	19	739	105

**Table:** Results for individual features when the candidate set contains all documents that match the query terms.

## Individual features with BFS

	success rate @ k			mean	median
	k=1	k=5	k=10		
BM25	9	49	61	15.6	5
TEXT	7	23	27	919	58
SUCC	8	22	31	465	31
PRED	8	26	32	232	27
SD	6	19	27	188	49

**Table:** Results for individual features when the candidate set contains documents that match the query terms and are found by a BFS traversal to depth 3 starting from the query node.

## Context-sensitive search — Results

	success rate @ k			mean	median
	k=1	k=5	k=10		
BFS_TPR	44	80	83	2.1	1
BFS_CPR	23	56	71	11.6	3
BFS_LPR	20	55	67	14.3	4
BFS_NPR	25	59	66	16.7	3
BFS_GBF	7	27	33	152	25
BFS_CBF	20	52	62	67.2	3
NOBFS_TPR	2	92	99	3.1	2
NOBFS_CPR	0	55	72	49.2	5
NOBFS_LPR	0	50	63	66.1	6
NOBFS_NPR	0	59	67	81.7	5
NOBFS_GBF	0	25	35	451	31

**Table:** Using RankSVM with different combinations of features. BFS and NOBFS tell if the candidate set was pruned by a BFS search starting from the query node. TPR, CPR and LPR stand for “true”, “cluster” and “landmark” pageranks, respectively. GBF (CBF) means that only the graph based (content based) features were in use.

## Qualitative results: “Jaguar”

Context 1 <b>Animal</b>	Context 2 <b>Computer Science</b>	Context 3 <b>Automobile</b>
Jaguar	Atari Jaguar	Jaguar XJ
Jaguar (car)	Mac OS X v10.2	Jaguar (car)
European Jaguar	Daimler Motor Company	Jaguar XK
Fender Jaguar	SEPECAT Jaguar	Jaguar XJS
Black panther	Mac OS X	Jaguar S-Type
Fender Bass VI	Jaguar X-Type	Jaguar E-Type
Pantherinae	Polymorphism (biology)	Jaguar X-Type
Panthera	British Leyland Motor Corp.	Jaguar XJ220
Jaguar XJ220	Royal Air Force	Daimler Motor Comp.
South America	HMS Jaguar (F34)	Jaguar AJ-V8 engine

## Qualitative results: “Cuckoo”

Context 1 <b>Animal</b>	Context 2 <b>Computer Science</b>	Context 3 <b>Film</b>
Cuckoo	Sopwith Cuckoo	One Flew over ...
Common Cuckoo	The Cuckoo's Egg	Jack Nicholson
Cuculus	Robert Morris (cryptogr.)	Ken Kesey
Coccyzus	Oberon-2	Academy Award
Yellow Cuckoo	Clifford Stoll	Louise Fletcher
Sumatran cuckoo	Bernardo Pasquini	Cuckoo clock
Clamator	Dictionary attack	1975 in film
Striped Cuckoo	Mike Muuss	1970s
Dideric Cuckoo	Computer insecurity	48th Academy Awards
Crotophagidae	Louise Fletcher	Milos Forman

- 1 Introduction
- 2 Approach
- 3 Experiments



Botev, C. and Shanmugasundaram, J. (2005).

Context-sensitive keyword search and ranking for xml.

In *8th International Workshop on the Web and Databases (WebDB)*.



Haveliwala, T. (2002).

Topic-sensitive pagerank.

In *Proceedings of the 11th WWW Conference*.



Joachims, T. (2002).

Optimizing search engines using clickthrough data.

In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.