J. Indian Inst. Sci., Sept.–Dec. 2003, **83**, 133–141
© Indian Institute of Science.

Short Communication

Context disambiguation in web search results

DEEPAK P[1]* AND SANDEEP PARAMESWARAN[2]
[1]Model Engineering College, Kochi 682 021, Kerala, India.
[2]IBM Global Services India Pvt Ltd, Embassy Golf Links, Bangalore 560 071, India.
Emails: deepakswallet@yahoo.com; sandeep_potty@yahoo.com; Phone: +91-9886019272.

Received on October 1, 2003; Revised on November 27, 2003.

Abstract

We commonly come across pages that are not of interest while searching the web. This is partly due to a word or words in the search query having different contexts, the user obviously expecting to find pages related to the context of interest. This paper proposes a method to disambiguate contexts in web search results.

Keywords: Context, referents, web search, disambiguation.

1. Introduction

Among the most important intentions of web usage is information retrieval and the most common activity pursued to achieve that is web searching. Precision is a very important yardstick to measure the quality of search engines. This paper presents a method to make web search results more customized in certain cases where the query terms have multiple senses.

Section 2 defines the problem at hand and the nature of the proposed solution. Section 3 deals with the different issues that concern such a solution. Section 4 reviews some related works of interest. Section 5 describes the approach used here. Section 6 presents the results of experiments using the approach. Section 7 lists the conclusions followed by a listing of references in Section 8.

2. The problem and the target

Search results often are poorly customized. If a user is interested in an entity A (it may be anything ranging from an incident, a product or a person) and there is more than one context for A, documents of all contexts are presented intermingled. From the results presented, the user has to discard the documents that do not belong to relevant contexts by making use of the automatically generated document extracts or descriptions that search engines provide with every page listed. Common examples include cases with multiple referents such as a search for a place, where places with the same name occur in more than one area.

*Author for correspondence. Permanent address: S-26, FACT Township, Udyogamandal 683 501, India.

There is a Kochi in India as well as in Japan. Michael occurs as the first name of many sports stars. A search for Bachchan would yield pages relating to Amitabh Bachchan, a legendary Hindi actor, as well as those on Abhishek Bachchan, a budding star in Hindi films.

Every user would have experienced this problem, but most of us take it for granted that it is our job to disambiguate the different contexts. This study aims at customizing web search results so that the pages relating to the same contexts (and same referents) may be presented together. This paper presents an approach whereby we can classify the pages and present to the user under different headings such as "pages on Kochi in the context: 'Kerala', 'India'", "pages on Kochi in the context: 'Japan'" etc., the former listing pages relating to Kochi in Kerala, and the latter relating to Kochi in Japan.

3. Factors related to the problem

3.1. *How it differs from the 'word sense disambiguation' problem*

Word sense disambiguation is an active field of research in natural language processing. It addresses a similar issue, identifying the sense of a word based on context. The word sense disambiguation community works on largely language-based techniques (such as verifying the grammatical form of the word). But we are interested in different contexts of the query (including multiple referents), rather than the sense. A search for "Mumbai blast" would yield pages related to the 1993 blasts as well as those related to the 2003 blast. Differentiating such contexts/referents would not be in the interests of the word sense disambiguation community. Moreover, our problem is not language specific and neither should the solution be. Thus the issue addressed here is inherently very different from the word sense disambiguation problem.

3.2. *Differences from 'text categorization' and neural net-based approaches*

The text categorization community and neural nets are concerned with similarity-based document classification. However, the problem here is to classify documents based on the context of the query used; not on the similarity. As the query is to be used as a parameter for classifying sets of documents, such approaches do not easily adapt to the problem due to their static structure. Furthermore, they do not provide the flexibility, to incorporate the search query as a special parameter.

3.3. *Need for yet another clustering algorithm*

The fundamental reason why many of the available clustering algorithms are unsuitable for this problem is the presence of an additional parameter, the search query. Most of the current clustering techniques take a set of documents and separate them into clusters of similar documents. Here we have to cluster the web pages in the context of the search query used. Similarity-based techniques would put the government reports on the Mumbai blasts of 1993 and 2003 in the same cluster, given the syntactical similarity between them as both are produced by the same government, probably using the same template. Speed is a major consideration as this service, when implemented as a meta-service, has to work on the fly between the production and presentation of results. Literature that deals with this precise problem or present solutions which possess such desirable features as listed above could not be found.

3.4. *Common context ambiguities*

Cases where there are different contexts for a query are abundant. The common ones include:

- Multiple referents:
  - Place names: Two places having the same name.
  - Names of people: There are many famous people with first names Michael.
  - Different events in the same place: 'Mumbai blasts' has many referents, two of them being the blasts of 1993 and also of 2003.
- Word sense ambiguities: These are much less important in the context of the web.

3.5. *Where to apply the solution*

A context disambiguator designed to solve the said problem would invariably have to work on a corpus or a collection of pages. The number of pages that the program gets to work on would provide more insight into the nature of the solution to be developed and the optimizations that can be done on it. One implementation would be to have the disambiguator embedded into the search engine itself. Search engines that implement the popular HITS[1] (hyperlink-induced topic search) algorithm [1] usually get a huge 1000-odd page corpus to work on. After splitting the whole set into different contexts containing a few hundreds of pages each, the HITS algorithm can be applied separately to each one of those sets. Another method would be to implement it as a meta-service which presents results from a search engine after disambiguation. The service would present the same results, differentiated into contexts. The disadvantage would be that this solution would get only 10–20 pages to work on, with only a few links between them. But such solutions would be fast, as they have to analyze only a handful of pages.

4. Related works of interest

Byrd and Ravin [2] describe the methodology used by IBM's Textract to identify and extract relations in text. It extracts relations between concepts extracted from the documents using the documents themselves. Another document clustering algorithm [3] builds co-reference chains for each document, uses them to collect sentences from each document and eventually creates summaries for each document. The summaries are used to cluster documents into collections. Both the works use language-dependent techniques, and are not suitable for inclusion of the search query as a parameter. Besides, Textract algorithm is computationally intensive.

[1]This is an algorithm which is used for web searching. It provides an ordered (by relevance) list of authoritative pages and hubs (pages with lists of links to authoritative web pages) on the search query when supplied with a collection of about 1000 pages collected from a text-based search for the query. It uses an iterative algorithm on the set of pages, aimed at boosting the scores of good hubs and authorities through the iterations.

A recent work [4] deals with distinguishing the real-world referent of a given name in context. It focuses on extracting biographical information from different documents using language-dependent methods. Although distinguishing real-world referents of a given name would be of interest in our problem, such name ambiguities form just one of innumerous possible ambiguities in web search results. Devising such specific methods for every possible kind of ambiguity would be impractical.

5. A context/referent disambiguation algorithm

The context/referent disambiguator presented builds a list of ⟨word, score⟩ tuples for each page and then, using the similarities between such lists, builds a graph with pages as nodes and undirected edges labeled with a measure of similarity between pages. Every densely connected component in the graph is then taken to represent a different context/referent for the search query used. This algorithm is oriented towards implementation as a meta-service, and is expected to perform well even if it has just 10–20 documents to work with.

5.1. *Part 1: Page-specific ⟨word, score⟩ list generation*

This approach takes each page, analyses it and builds a list of ⟨word, score⟩ tuples for it. Those keywords that are specific to the context/referent to which the page belongs should be given high word scores. The more general inter-context keywords should end up with low scores. The algorithm described here uses proximity to the search query and frequency as the parameters to the score computation function. The algorithm makes no attempt to identify inter-context keywords (to suppress their scores) as such a procedure would be highly non-trivial and error-prone. The algorithm is given below:

```
Procedure ListGen(Page p)
{
   Score of every word in page p = 0;
   For every word, W in page p
   {
      For every occurrence w of W in page p
      {
         (freq_score of W) += 1;
         (prox_score of W) += (THRES – least number of words intervening
         between w and a word in search query in page p) + BOOST;
      }
      total_score of W = freq_score of W + prox_score of W;
   }
   Normalize word scores such that the (sum of total_score of every word in p = a limit);
   Make a list of <word, total_score> tuples containing an entry for each word in p;
}
```

The freq_score holds the frequency of a word in page p. Prox_score holds the score of each word due to the proximity of its occurrence to the words in the search query. The scores are normalized so that they add up to an upper limit granting each page the same influence in the next stage of disambiguation. BOOST can be set to a value based on the relative weighting to be given to proximity and frequency-based scores. The ⟨word, score⟩ tuples created here are used in subsequent stages.

5.2. *Part 2: Document clustering*

This builds a huge list of unique words, listing every word occurring in the list of at least one page. Each page is represented as a vector with the *i*th element holding the score for the *i*th word in the list for that page. A graph is created with the pages as nodes and undirected edges[2] between two pages labeled with the dot product[3] of the vectors of the two pages. All edges having labels below a threshold are pruned so that densely connected subgraphs become isolated connected components. The pages in each such connected component are taken as belonging to a separate context.

```
Procedure DocCluster (pages, each with an associated list of ⟨word, score⟩ tuples)
{
   Create a list of n unique words, ⟨word[1], word[2] … word[n]⟩ such that a word, w
      occurs in the list if it has an associated score for at least one page in the
      collection;
   Represent page p as a vector, ⟨a[1], a[2],… a[n]⟩ where a[i] assumes the value,
      where ⟨word[i], k⟩ is a tuple of page p or
      0 if there is no tuple with word[i] for page p
   Create an undirected graph[4] with pages as nodes and the edges between any two
      pages labeled with the dot product of the vectors of the two pages,
      scaled by an appropriate factor;
   Prune every edge in the graph where the weight is below a THRESHOLD;
   Each densely connected subgraph of the resulting graph represents
      a different context;
   }
```

Fixing the THRESHOLD at a low value may result in merger of contexts/referents, and thus poor ambiguity resolution. A high value would invariably break up contexts/referents which seem logically coherent to the human, into further smaller contexts.

5.3. *Part 3: Identification of context-keywords to describe the context/referent*

After identifying the different contexts/referents, we describe them to the user by a set of keywords. The algorithm creates a context-wide list of words, each word, having associated with it a score, that is computed by taking the sum of scores for the word in each page of the context group. The words with the highest scores are taken to be the context keywords.

Procedure context-keywords (collection of pages of a context)

[2]Undirected edges: Edges that do not have an orientation. They do not have a source or destination vertex, and they just connect the two vertices.

[3]Dot product: It is the scalar product of two vectors and is defined as the sum of the products of corresponding components. It is denoted by a dot. For example, ⟨a, b, c⟩.⟨d, e, f⟩ = d + be + cf.

[4]Undirected graph: It is a graph with only undirected edges connecting the vertices.

```
{
   Create a list of n unique words, ⟨word[1], word[2]… word[n]⟩ such that a word, w
      occurs in the list if it has an associated score for at least one page in the
      collection;
   Associate each word with a score initialized to 0;
   For (each page, p)
   {
      Score of word[i] += k where there is a tuple <word[i], k> for page p
   }
   Take the words with the highest scores as the context-keywords for the context in
      question;
}
```

5.4. *Related issues*

The score computation part assigns scores to words based on their relative frequency and proximity to the search query words. It is based on the assumption that context/referent-specific keywords occur close to occurrences of search query words. No weighting has been given to link-based information as the target of the link would be unknown in most cases. Thus it treats hypertext just as text documents. Finding the densely connected disjoint sub-graphs should be a complex and time-consuming algorithm, although usage of such an algorithm would certainly provide better results than the approach used here. If no isolated connected components are found by part 2 of the algorithm, we may conclude that there is only one context for the whole set of pages.

6. Test results

The algorithm as given above was implemented and tested on many collections: a collection of pages on the search query ‘Kochi’ containing pages on the places with the said names in India and Japan, another collection which contained pages on Joe Jackson and Michael Jackson collected using the search query, ‘Jackson’ and various other collections. For the tests, BOOST and THRES were given values of 4 and 10, respectively. The graphs obtained were processed by first pruning edges having weights less than 20% of the weight of the strongest edge. Then more edges were pruned from the remaining graph with the threshold for pruning edges chosen as half the weight of the strongest edge or the value that preserves only 20% of edges, whichever was lower.

6.1. *The ‘Kochi’ and ‘Jackson’ collections*

The ‘Kochi’ collection contained five pages each, on the Japanese city as well as on the Indian city. As can be seen from Fig. 1, there is just one weak intercontextual link, which was seen to be due to a general intercontext keyword, ‘page’. The pendant page was found to contain little information. The pages on Kochi, India, form a densely connected subgraph with strong edges and so do the pages (except one) on Kochi, Japan. The keywords for the Indian context were ‘Kerala’ (Kochi is in ‘the state of Kerala’) and ‘Cochin’ (former name of Kochi), those for the Japanese city being ‘Japanese’ and ‘Japan’.

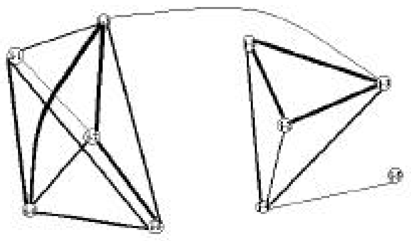

FIG. 1. The 'Kochi' collection graph. Nodes with labels starting with 'i' represent pages on Kochi, India, and those starting with 'j' represent 'pages on' Kochi, Japan. The thicknesses of the edges represent the magnitude of the labels of the edges.

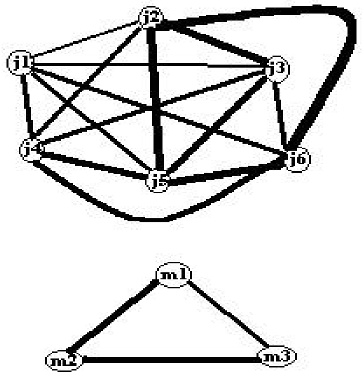

FIG. 2. The 'Jackson' collection graph. Nodes with labels starting with 'j' represent pages about Joe Jackson and those with labels starting with 'm' represent 'pages on' Michael Jackson. The thicknesses of the edges represent the magnitude of the labels of the edges.

The 'Jackson' collection contained nine pages, six pages on 'Joe Jackson' and the remaining three on 'Michael Jackson'. As can be seen from Fig. 2, the pages from each context form a dense clique themselves. 'Joe' and 'Michael' were identified as the top keywords for the 'Joe Jackson' and 'Michael Jackson' collections, respectively.

6.2. *Tests on miscellaneous collections*

This section presents tests performed on collections obtained from a search engine called Google (http://www.google.com) for various queries. Pages listed as the top 20 (in the results) were downloaded and subjected to the context disambiguation algorithm. The queries used were chosen at random (not like the previous tests where the desirable contexts were known before hand). Given below are the results on six collections, each labeled by the search query used, a random sample from the 21 collections tested.

'*Birthday*' *collection:* This collection contained 17 pages. The algorithm identified one context of 10 pages, all with online greeting pages or birthday parties. But two pages among the excluded ones were found worthy of inclusion in the context, on manual inspection. The other pages were those dealing with the birthday celebrations of various political or cultural leaders, which bore no similarity among them.

'*Compilers*' *collection:* Two contexts were identified by the algorithm on this 11-page collection, one containing three pages giving information about 'what compilers are' and another two pages providing lists of different compilers available. All other pages were homepages of different compilers which evidently cannot be clustered into a context. The algorithm was seen to have performed very well in this collection.

'*Dijkstra*' *collection*: This collection contained six pages. A context of four pages was identified, all of which contained biographical sketches on the life of the scientist. The excluded pages contained a description of Dijkstra's algorithm and an extract from the scientist's lecture against the usage of 'goto' statements.

'*Kerala*' *collection:* This collection contained 12 files. 'Kerala' is the name of a state in India. No edge survived the edge-pruning phase. Upon analysis of the collection, it was found that the pages contained the homepages of Kerala police, Kerala High Court, Kerala Tourism Department, Kerala government, etc. which bore no semantic similarity. Thus the algorithm did perform well even in this extremely adverse situation.

'*Telgi*' *collection:* Telgi is the name of the prime-accused in a fake stamp-paper case in India. The collection contained 17 files, in which the algorithm identified two contexts of six pages each. One contained newspaper reports on a rumor as to whether Telgi was injected with HIV when in police custody. The other context contained pages on the government stand on the Telgi issue. The excluded pages were reports on various minor details of the Telgi case, which on manual inspection, were found to bear no semantic similarity among them as well as with the identified contexts. Thus the algorithm is seen to have performed well, although not 'perfectly'.

'*Police*' *collection:* This collection contained 19 pages obtained by the query 'Police' on pages relating to 'Kochi'. A single context of six pages was identified, of which five pages contained reports on crimes in Kochi and one with a list of police officers, which had strong edges to each of the other five pages, although it doesn't seem to be worthy of being included in a context of crime reports. The links were due to a high frequency of investigating officers' names in crime reports. Such cases where a 'different' page holds a community of cohesive pages together may be worthy of further investigation.

7. Conclusions and future work

It can be seen from the tests that the algorithms perform very well even when they have very little information to work with. The obvious concern would be as to how the algorithm would scale with increase in the number of pages in input. A meta-service implementation need process only the first 20–30 result pages at a given time and can defer the disambiguation of other pages to a time when they are actually required. Thus, we are justified in ignoring the problem of scalability if the algorithm is to be implemented as a meta-service.

Future work may be directed towards formulating optimizations when the disambiguator is implemented within a web search algorithm. The optimizations may be formulated to exploit the larger corpus and any link-based information that can be used. The performance of the algorithms on very broad topic queries such as 'union' whose context ranges from 'rugby unions' to 'c unions' and in cases of word sense ambiguities, have to be investigated. Such cases are different from the cases discussed here, but investigations as to how the algorithms presented here work on such problems, might provide clues on what the nature of solutions for such varied problems should be.

References

1. J. M. Klienberg, Authoritative sources in a hyperlinked environment, *J. ACM*, **46**, 604–632 (1999), URL: http://citeseer.nj.nec.com/kleinberg99authoritative.html.

2. R. J. Byrd and Y. Ravin, Identifying and extracting relations in text, *Proc. 5th Jt Conf. on Information Sciences,* JCIS2000, pp. 207–210 (2000).

3. A. Bagga and A. W. Biermann, A methodology for cross-document coreference, *Proc. NLDB99,* Klagenfurt, Austria (1999).

4. G. S. Mann and D. Yarowsky, Unsupervised personal name disambiguation, *Proc. Seventh CoNLL Conf.*, Edmonton, May–June 2003, pp. 33–40.