Clustering Web Search Results into Contexts

Sandeep Parameswaran,
IBM Global Services India Pvt. Ltd., Bangalore, India
psandeep@in.ibm.com

Deepak P,
Model Engg: College, Kochi, India
deepak-p@eth.net

Introduction

It is a common experience while web searching that one gets to see pages that are not of interest. Partly these are 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 for disambiguating contexts in web search results. The algorithm consists of 2 parts as detailed below.

The Approach

Part 1: Page Specific <Word, Score> List Generation: This approach takes each page, analyses it and builds a list of <word, score> tuples for each page. Those keywords that are specific to the context/referent to which the page belongs should invariably be given high word scores. The more general inter-context keywords should end up with low scores. The proximity to the words in the search query would be an obvious parameter to the score computation function.

Table 1. Page-specific Word-Score list generation

```
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;
}
```

Part 2: Document Clustering: This builds a huge list of unique words, with every word occurring in the list of at least one page listed. 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 between two pages labeled with the dot product 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 different context.

0-7803-8482-2/04/$20.00 ©2004 IEEE.

Table 2. Document Clustering

```
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 atleast one page in the
   collection;
  Represent page p as a vector, <a[1], a[2],... a[n]> where a[i] assumes the value,
        k 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 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;
}
```

Having identified the different contexts in the search listings, we can apply algorithms to find the keywords for each context, so that results can be displayed as sets, ' results in the context : list of words in the context ' for each context.

Conclusions

It was seen from the tests that the algorithms perform very well even when they have very little information to work with. Thus implementing context disambiguation as a meta-service may be an interesting experiment. The obvious concern would be as to how the algorithm would scale with increase in the number of pages in input. Search engines usually provide results in pages of 10 (or more) results. Thus, a meta-service implementation of the disambiguator need process only 2-3 result lists, and thus 20-30 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 altogether if the algorithm is to be implemented as a meta-service. But in cases where the algorithm is to be embedded into the search algorithm, such issues may assume greater significance.

Future Work

Future work may be directed towards formulating possible optimizations when the disambiguator is implemented within a web search algorithm (in the search engine architecture itself). The optimizations may be formulated to exploit the larger corpus. Further link based information may also be used in such cases. The keyword identification algorithm can also be improved considerably making use of principles and results from computational linguistics. The performance of the algorithms in very broad topic queries such as "union" whose context ranges from "rugby unions" to "c unions" and in cases of word sense ambiguities and other issues, not dealt with in the current paper, have to be investigated. Such cases are clearly different from the cases discussed through the bulk of the paper, but investigations as to how the algorithms presented here work on such problems, might provide valuable clues on what the nature of solutions for such varied problems should be.