

# Automatically Constructing Descriptive Site Maps

Pavel Dmitriev<sup>1</sup>, Carl Lagoze<sup>1</sup>

<sup>1</sup> Cornell University, Department of Computer Science,  
Ithaca, NY, 14853, USA  
{dmitriev, lagoze}@cs.cornell.edu  
<http://www.cs.cornell.edu>

**Abstract.** Rapid increase in the number of pages on web sites, and widespread use of search engine optimization techniques, lead to web sites becoming difficult to navigate. Traditional site maps do not provide enough information about the site, and are often outdated. In this paper, we propose a machine learning based algorithm, which, combined with natural language processing, automatically constructs high quality descriptive site maps. In contrast to the previous work, our approach does not rely on heuristic rules to build site maps, and does not require specifying the number of items in a site map in advance. It also generates concise, but descriptive summaries for every site map item. Preliminary experiments with a set of educational web sites show that our method can construct site maps of high quality. An important application of our method is a new paradigm for accessing information on the Web, which integrates searching and browsing.

## 1 Introduction

Recent research indicates that the Web is continuing to grow rapidly. However, the number of web sites did not increase much over time [8]. Thus, the growth is mostly due to the increase in the number of pages on web sites. According to the OCLC web survey [7], the average number of pages on a public web site already was 441 in 2002. Such increase in complexity of web sites inevitably makes them more and more difficult to navigate.

It is not only the growth in the number of pages that complicates navigation on a web site. The dominance of search engines as the primary method of accessing information on the web discourages web site developers from paying enough attention to making their web sites easy to navigate. In addition, developers employ tricks to raise the ranking of their sites by search engines, making navigation even more difficult [9].

The traditional way to help users with navigating through a web site is a site map. A site map is a web page that contains links to all the main sections of the web site, and, possibly, a concise description of what each section is about. A site map is usually created and maintained by the web site administrator.

In reality, however, many web sites do not have site maps. And those that do usually do not have descriptions of sections, and are often outdated. This is not surpris-

ing, since site maps have to be created and maintained manually. Thus, there is a strong need for a technique capable of automatically generating accurate and descriptive site maps.

The task of automatically creating a site map can be thought of as consisting of several steps. First, important sections on a web site must be identified<sup>1</sup>. Then, the sections must be combined into a hierarchy, taking into account user-defined constraints, such as the desired depth of the site map, maximum number of items in the site map, etc. Next, anchor text must be generated for every item. Finally, summaries for the sections need to be generated, which accurately describe contents of the sections.

There are several important problems one needs to solve in order to successfully implement the above steps. There are a variety of structural and content features of web pages that determine important sections on web sites, such as content, anchor text, link patterns, URL structure, common look-and-feel. Simple heuristics that use only one or two of these features are not sufficient to identify web site sections [3]. A more complex approach is required, which would combine all these diverse features in an appropriate way. Combining the sections into a hierarchy is difficult, too, since the link structure of the site usually allows for multiple ways to do it, and user-defined constraints may require merging some of the sections. Finally, generating the titles and summaries must be done in such a way that the user has enough information to decide whether he or she should navigate to the section, while keeping the summaries small to allow skimming through the overall structure of the web site.

In this paper we propose a method for automatically constructing descriptive site maps. In contrast to the previous work, which used a heuristic-based approach [6], our work is based on a new semi-supervised clustering algorithm [3]. Given several sample site maps, a learning algorithm decides on the best way to combine multiple diverse features of web pages into a distance measure for the clustering algorithm, which is used to identify web site sections. Note that, in contrast to [6], our approach does not require specifying the number of clusters/sections in advance – the clustering algorithm decides on the optimal number of clusters automatically. The resulting clusters are then processed to identify leader pages, and iteratively merged into a hierarchy that satisfies user-defined constraints. Finally, titles and summaries are generated for site map items using multi-document summarization techniques.

An important application of our algorithm is a new paradigm for accessing information on the Web, which integrates searching and browsing. Currently, when the user clicks on a link on a search engine results page, they get to a page that is totally out of context. There is generally no way to find out the function of this page in the web site as a whole, or navigate to related pages or other sections of the site. This is because web sites are generally designed with the assumption that the user will always start navigation from the root page of the site. Search engines, on the other hand, bring users to a “random” page on the site. To solve this problem, search engines could use our algorithm to build a site map for every site in their index. This

---

<sup>1</sup> Note that this is different from simply identifying important pages. Important pages are the root pages of the sections; however, an assignment of each regular page to a root page must be computed as well, since it is essential for generating a high quality summary of the section.

can be done offline and in incremental fashion. Then, the site map, as well as various statistics about the site, can be presented to users when they navigate from search engine to the page.

The rest of the paper is organized as follows. The next section presents an overview of related work. Section 3 describes our algorithm for constructing descriptive site maps. Section 4 discusses our experiments, and section 5 concludes the paper.

## 2 Related Work

Research relevant to our work can be divided into two categories: recovering internal structure of a web site and automatic site map construction, and multi-document summarization. Below we give an overview of work in each of these areas.

**Recovering Web Site Structure.** Eiron and McCurley [4] propose a heuristic approach to identifying important sections on a web site, which they call *compound documents (cDocs)*. Their approach relies heavily on the path component of URL. Every outlink on a page can be classified as being up, down, inside, or across link, with respect to whether it links to a page in an upper, lower, same, or unrelated directory. Eiron and McCurley observe that, since cDocs are usually authored by a single person and created over a short period of time, they tend to contain the same (or very similar) set of down, inside, and across links. Another heuristic they use is based on an observation that outlinks of the members of a cDoc often have the same anchor text, even if their targets are different. Finally, they assume that a cDoc cannot span across multiple directories, with the exception of the leader page of a cDoc being in one directory, and all other pages being in a subdirectory. These heuristics are manually combined to identify cDocs on web sites.

Li et. al. [6] propose a heuristic approach to identifying *logical domains* – information units similar to cDocs – on web sites. Their heuristics are based on observations about what leader pages of logical domains tend to look like. They use a number of heuristics exploiting file name, URL structure, and link structure to assign a leader score to every page on the site. Top  $k$  pages are picked as leader pages of logical domains, and every other page is assigned to one of the leader pages based on the longest common substring of the URL, making adjustments to make sure that all pages of a logical domain are accessible from the leader page.

The approach of [6] was also applied to automatic site map construction [2]. First, logical domains are identified using the approach described above. Then, for every pair of parent logical domain leader page and child logical domain leader page,  $l$  pages best describing the association between the two leader pages are chosen using a PageRank style algorithm. The site map is then constructed by finding shortest paths between all selected pages. The approach does not include generating anchor texts or summaries.

Both of the above approaches are based on a number of pre-defined heuristics, and thus fail to account for a variety of content, style, and structural conventions existing in different communities on the Web. In addition, the latter approach requires specifying the values for parameters  $k$  and  $l$  in advance.

Trying to overcome the limitations of the methods mentioned above, we proposed in our previous work [3] a new approach to finding compound documents. This work uses an extension of that approach to identify important sections on web sites. We describe our approach in detail in section 3.

**Summarizing Multiple Documents.** The task of multi-document summarization is to produce, given a set of documents, a coherent summary of a specified size describing the contents of all documents at once. The approaches to multi-document summarization can be split into two classes: those that produce summaries consisting of whole sentences (or large sentence fragments) extracted from the original documents, and those that produce summaries consisting of new automatically generated sentences based on semantic information extracted from the documents. Although recently there has been some work in the latter area, the proposed methods are computationally intensive, and, to our knowledge, do not perform significantly better than the approaches from the more simple former class. Thus, we only concentrate on the former class here.

Most of the approaches of the former class consist of three main stages. On the *feature extraction* stage, for every sentence a number of features are extracted relevant to estimating significance of the sentence; on the *sentence ranking* stage, every sentence is assigned a significance score based on the extracted features; on the *summary generation* stage a summary is generated from highly ranked sentences, paying particular attention to avoiding adding redundant information to the summary. Due to space limitations, we do not discuss how each of these stages is implemented in existing systems. An interested reader is referred to the DUC web site, <http://duc.nist.gov>, for more information.

Our approach to generating summaries follows the standard framework outlined above. We describe the particular methods used in section 3.

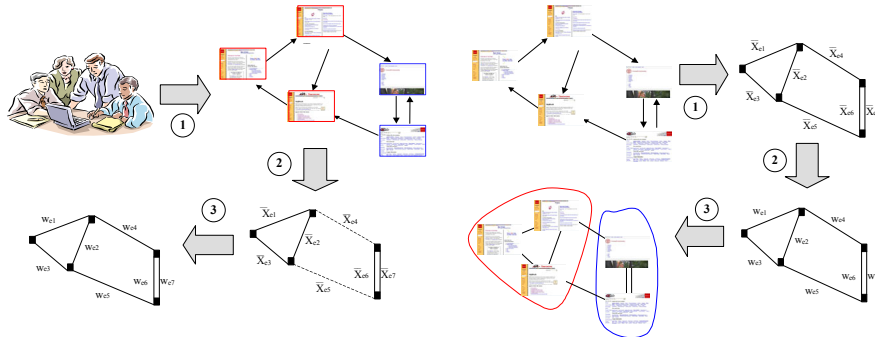
### 3 Constructing Descriptive Site Maps

Our approach to constructing descriptive site maps consists of three stages. On the *compound documents identification* stage, a web site is split into a set of cDocs, each cDoc representing an important section of the site; on the *site map construction* stage, the cDocs are combined into a hierarchy to produce a site map according to user-defined criteria; on the *anchortext and summary generation* stage, anchortext is generated for every item in the site map, and a summary is extracted for every leaf item. We describe each of these stages in detail in the subsequent sections.

**Finding Compound Documents.** This stage is implemented using the system for automatic identification of compound documents we developed in our previous research [3], extended with additional functionality. We represent a web site<sup>2</sup> as a graph with nodes corresponding to pages, and edges corresponding to hyperlinks. The process of finding cDocs consists of two phases.

---

<sup>2</sup> In this paper, by a *web site* we mean all pages under the same domain name.



**Fig. 1.** Training phase (left), and working phase (right) of the process of identifying compound documents. Solid edges are positive, and dotted edges are negative training examples.

On the *training phase* (fig.1, left), a human labels cDocs on several web sites. Then, a vector of features<sup>3</sup> is extracted for every hyperlink, and a logistic regression model is trained to map vectors of feature values to weights on the corresponding edges of the graph. On the *working phase* (fig.1, right), given a new web site, we repeat the process of extracting the features from the pages, and transforming them into vectors of real values. Then, we use the logistic regression model to compute weights on the edges. Finally, a graph clustering algorithm is applied to the weighted graph. The clusters are the cDocs we are looking for.

We make a few notes about our clustering algorithm. The algorithm is an extension of the classical agglomerative clustering algorithm. It starts with every page being a singleton, and then repeatedly merges the clusters, picking edges one by one in the order of decreasing weight. However, there is a notable difference from the classical case. Our algorithm usually stops before all the pages have been merged into a single cluster, since it may decide not to merge two clusters under consideration on a particular step, based on the average weights of edges within the clusters, and the weight of the edge connecting them. The threshold controlling this process is learned during the training phase.

In this work, we extended the clustering algorithm described above with the ability to satisfy the *alldiff* constraint, which requires, for a given set of pages, that no two of them belong to the same cluster. To satisfy it, every time we want to merge two clusters, we have to check that the merge is allowed. Note that this only introduces a constant overhead on every step of the clustering process<sup>4</sup>. We also note that this approach can be applied to enforce alldiff constraint in the classical agglomerative

<sup>3</sup> The features we use are content similarity, anchor text similarity, title similarity, number of common inlinks, number of common outlinks, number of outlinks with common anchor text (the outlinks linking to different pages, but having identical anchor text), and whether the hyperlink points to the same directory, upper directory, lower directory, or other directory with respect to the path component of the URL.

<sup>4</sup> To see this, let every page keep a pointer to its clusterID, and let every clusterID have a mark indicating whether it contains a page from the alldiff set. Then, to check the constraint we simply check whether both clusterIDs are marked, and, when two clusters are merged, we only need to update the clusterID and the mark fields.

algorithm as well, making it produce, instead of a single cluster, the number of clusters equal to the number of pages in the alldiff set.

We apply alldiff constraint to all pages with filenames `index.<ext>` or `default.<ext>`. The intuition (confirmed experimentally) is that two pages having such filenames are unlikely to belong to the same cDoc. Moreover, as we discuss in the next section, such pages are very likely to be leader pages of their cDocs.

**Constructing Site Maps.** To construct a site map we need, for every cDoc, to (1) find a *leader page*, i.e., the most natural entry point into the cDoc, (2) combine all cDocs on a site into a single hierarchy, and (3) apply user-defined constraints to the hierarchy to produce a site map satisfying the user's requirements. Below we describe how each of these steps is implemented.

Identifying leader pages in cDocs is important, because these are the pages that items in the site map should link to. We use two simple, but reliable heuristics to identify the leader page in a cDoc<sup>5</sup>. First, the filename of every page is examined. If a page with filename `index.<ext>` or `default.<ext>` is found, it is chosen to be the leader page. Note that there can be at most one such page in a cDoc, due to the alldiff constraint described earlier. Second, for cDocs that do not contain pages with the above filenames, the page with the greatest number of external inlinks (i.e., inlinks from other cDocs) is chosen to be the leader page.

After the leader pages are identified, the cDocs are combined into a hierarchy to produce a site map. The cDoc containing the root page of the web site is taken to be the root node of the site map. Then, all outlinks from the pages of the root cDoc are examined, and all cDocs whose leader pages these outlinks point to are taken to be the descendants of the root node in the site map. New nodes are then examined in the breadth-first order, and processed according to the same procedure.

Finally, user-defined constraints are applied to the site map produced on the previous step. Currently, the user can specify three types of constraints in our system: the minimum number of pages a cDoc must have to be present in a site map, the maximum depth, or number of levels in the site map, and the maximum number of leaf items in the site map.

These constraints allow the user balance the comprehensiveness and readability of the site map. All three constraints are enforced in a similar manner, by iteratively traversing the site map in a depth-first order, and merging the nodes that violate the constraints into their parent nodes. Table 1 shows the values for these constraints used in our experiments.

**Generating Summaries and Anchortexts.** To generate summaries and anchortexts, we follow the standard multidocument summarization framework described in section

---

<sup>5</sup> Our original plan was to train a classifier using the various features that could potentially be useful in determining the leader page of a cDoc. However, experiments with the heuristic approach showed that there is no need for that. Indeed, the heuristic approach showed 100% accuracy at identifying leader pages on all cDocs from 5 randomly chosen sites from our dataset.

2, i.e., the following three steps are performed: feature extraction, sentence ranking, and summary and anchortext generation.

On the feature extraction stage, anchortexts, titles, and content are extracted for every site map item (referred to as *texts* in the remainder of the paper). In addition, the centroid, as well as  $k$  most frequent keywords are extracted for every text. After that, every text is processed with Automated English Sentence Segmenter<sup>6</sup> to identify sentence boundaries. Filtering is then applied to mark sentences not likely to be useful for the summary (the ones containing email addresses, phone numbers, copyright statements, etc.). These sentences will only be considered if a summary of the required size could not be constructed from unmarked sentences.

For sentence ranking, we experimented with two approaches: rank the sentences according to the similarity of the sentence to the centroid of the text, and rank the sentences according to the similarity of the sentence to the  $k$  most frequent keywords of the text.

Once the ranking is computed, we can generate anchortexts and summaries for our site map. We tried three approaches to anchortext generation: the highest ranked sentence from the text consisting of anchortexts of the leader page of the item, the highest ranked sentence from the text consisting of titles of all pages of the item, and the title of the leader page of the item.

```
Filter out all sentences of length more than a threshold;
Pick the highest ranked sentence and include it in the summary;
Until (the summary contains the desired number of sentences) do {
    Pick the next highest ranked sentence;
    If ((similarity of the sentence picked to the existing summary <  $t_1$ )
    && (its similarity to the centroid/keywords >  $t_2$ ))
    Then include the sentence in the summary;
    Else drop the sentence and continue;
}
```

**Fig. 2.** Summary generation algorithm

For summary generation, we tried eight approaches, which differ in the type of text they used, and in the ranking method. We applied the two ranking methods mentioned above to the text of the leader page, text of all pages, and text of first sentences of all pages. In addition, we tried a hybrid approach, taking text of the leader page, and using centroid and keywords of all pages to rank the sentences.

The summary generation algorithm used in all these cases is shown on Figure 2. It has a number of parameters. The desired number of sentences in the summary is a parameter set by the user, which lets them control the size of the generated summary. The sentence similarity threshold,  $t_1$ , is used to avoid including in the summary two sentences that are very similar. The higher the value of this threshold, the stricter the requirement is that a new sentence included in the summary must be substantially

---

<sup>6</sup> <http://www.answerbus.com/sentence/>

different from the current summary. Finally, the sentence relevance threshold,  $\tau_2$ , ensures that the sentence picked, while being substantially different from the rest of the summary, is still relevant to the centroid, or contains some of frequent keywords. Table 1 shows all parameters used by our algorithm.

**Table 1.** Parameters of the algorithm

Name	Description	Value
Min. # pages in an item	The minimum number of pages a site map item has to contain	3
Max. depth of a site map	The maximum height of a site map tree	3
Max. # leaf items	The maximum number of leaf items in a site map	30
Max. summary length	The maximum # sentences in the summary of a site map node	3
Max. sentence length	The maximum # characters a sentence may have to be included in the summary (longer sentences can still be included, but must be cut up to 150 characters)	150
Sentence similarity threshold	The largest similarity a new sentence may have to the existing summary to be added to the summary	0.5
Sentence relevance threshold	The smallest similarity a new sentence must have with the centroid/keyword list to be added to the summary	0.1
# keywords	Number of words used to generate the list of most frequent terms	5

## 4 Experimental Results

For the purpose of evaluating our algorithm, we used the same dataset that we used in [3] to evaluate our system for finding compound documents. This dataset consists of 50 web sites on educational topics. However, since many of the web sites in that dataset were too small to make site map construction for them interesting, we ended up using only 20 web sites. We are currently working on preparing a significantly larger dataset, which we will use for a more thorough evaluation.

It is difficult to formally evaluate the quality of site maps, since there are too many subjective factors involved in deciding how good a site map is. Therefore, we conduct a separate evaluation of every component of the system. Figure 4 can give the reader an idea of what a typical site map generated by our system looks like.

**Compound documents identification.** Since the quality of the system for identifying compound documents was extensively evaluated in [3], and we use the same dataset to evaluate our system, we do not repeat the details of the evaluation here. The key results of [3] were that the system could identify correctly most of the compound documents, a small number of training sites (typically 6) were enough to train a good logistic regression model, and the system was relatively insensitive to the choice of training examples. Here we used 10 sites for training, and the other 10 for testing.

**Leader page selection.** As we mentioned in section 3, our heuristic approach to leader page identification produced 100% accuracy, when evaluated on all the compound documents from the 5 evaluation sites.

- [The Prime Pages \(prime number research, records and resources\)](#)
  - [Lists of primes](#)
    - [Single primes](#)  
**Keywords:** **prime, certificate, primes, bit, random.**  
 A 36,007 **bit** "Nearly **Random**" **Prime** A 36,007 **bit** "nearly **random**" **prime** This 10,839 digit **prime** does not have a nice short description!... While new records with cyclotomy are being boiled, I have another kind of large **prime** for you: a 36007 **bits** almost **random**, proved **prime**... Mersenne Glossary **Prime** Curios!...  
[http://www.utm.edu/research/primes/lists/single\\_primes/](http://www.utm.edu/research/primes/lists/single_primes/) - 7pages
    - [Lists of small primes \(less than 1000 digits\)](#)  
**Keywords:** **primes, first, digits, prime, twin.**  
 Move up one level] From other sites All the **primes** below 100,711,433 (5.8 million **primes**) All the **primes** below 2,000,000,009 (98 million **primes**)... **Prime** Lists FAQ e-mail list Titans **Prime** Links Submit **primes** What is small? Depends on your context, but since this site focuses on titanic **prime**... Lists of small **primes** (less than 1000 **digits**) Lists of small **primes** (Another of the **Prime** Pages' resources) Home Search Site Largest The 5000...  
<http://www.utm.edu/research/primes/lists/small/> - 4pages
    - [Primes just less than a power of two](#)  
**Keywords:** **prime, bits, primes, 69, 45.**  
 Pages: 8-100 **bits**, 101-200 **bits**, 201-300 **bits**, 301-400 **bits**. n ten least k's for which 2n-k is **prime**.... **Prime** Lists FAQ e-mail list Titans **Prime** Links Submit **primes** Here is a frequently asked question at the **Prime** Pages: I am working on an algorithm and... **Prime** Lists FAQ e-mail list Titans **Prime** Links Submit **primes** When designing algorithms, sometimes we need a list of the **primes** just less than a power...  
<http://www.utm.edu/research/primes/lists/2small/> - 5pages
  - [Modular restrictions on Mersenne divisors](#)  
**Keywords:** **prime, mod, p-1, theorem, proof.**  
 Let p be a **prime** and a any integer, then  $ap = a \pmod{p}$ ....  $1 \pmod{p}$ ). Finally, multiply this equality by **p-1** to complete the **proof**.... Let p be a **prime** which does not divide the integer a, then  $ap-1 = 1 \pmod{p}$ ....  
<http://www.utm.edu/research/primes/notes/proofs/MerDiv.html> - 5pages
  - [Proofs that there are infinitely many primes](#)  
**Keywords:** **prime, primes, primality, theorem, test.**  
**Prime** Lists FAQ e-mail list Titans **Prime** Links Submit **primes** Euclid may have been the first to give a proof that there are infinitely many **primes**.... **Prime** Page References **Prime** Page References (references for the **Prime** Pages) Home Search Site Largest Finding How Many?... Notice that different a's can be used for each **prime** q.) **Theorem** 2 can be improved even more: if F...  
<http://www.utm.edu/research/primes/notes/proofs/infinite/index.html> - 17pages

**Fig. 3.** A fragment of a site map for [www.utm.edu/research/primes/index.html](http://www.utm.edu/research/primes/index.html) built using text of all pages and keyword-based ranking

**Site map construction.** We compare the site map produced from automatically identified compound documents to the one produced from gold-standard compound documents. Out of the 5 evaluation sites, site maps for 2 sites were identical, site maps for another 2 sites were very similar, with differences only in a few leaf-layer nodes, and for one of the sites the site map produced from the automatically identified cDocs was substantially different (and of less quality) than the one produced from the gold-standard cDocs. The difference, however, could be minimized with setting different

values for the maximum number of allowed leaf nodes and maximum depth of the site map<sup>7</sup>. Overall, the performance of the system on this step, though not as good as it is on the previous step, is still very good.

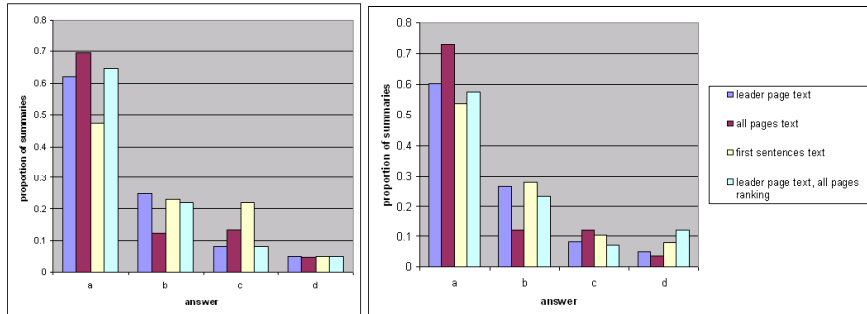
**Anchortext generation.** Contrary to our expectations, the approach using anchortexts performed rather poorly, and the two approaches using titles showed reasonably good (and in most cases identical) results. It turned out that anchortexts within a single site often do not contain descriptive information at all, referring with the same word, such as “introduction”, or “index” to almost all leader pages. In many cases anchortexts repeat titles of the pages they point to, and often they contain pictures, rather than words. This is particularly interesting because anchortexts have been found very effective in web information retrieval [1], as well as information retrieval in intranets [5], exactly because of their descriptive properties: the home page of Google does not contain the phrase “search engine”, but anchortexts often do. We attribute the difference in the results between our own and previous experiments to the difference in the nature of the datasets. Previous work studied the public web, or a large corporate intranet as a whole. To our knowledge, this work is the first to analyze anchortexts within individual web sites. Our results suggest that anchortexts within the same web site are mostly useless. This suggests that search engines might be able to improve the quality of their results by ignoring all anchortexts that come from the same site as the page itself.

**Summary generation.** Again, 5 sites were picked randomly, which resulted in 58 items with summaries. The algorithm described in section 3 was used to generate summaries in eight different ways. Then, we evaluated the summaries generated using each of the methods according to the following question: “does the summary give the user a good idea of the topic/function/goal (depending on the type of the item) of the site map item it describes?” The possible answers are: (a) gives a very good idea; (b) gives a pretty good idea; however, some important information is missing; (c) gives a rough idea of it; (d) there are some clues of it in the summary, but they are not very easy to see; (e) the summary contains no useful information<sup>8</sup>. The results are shown on Figure 4.

---

<sup>7</sup> In general, we found that the optimal values for the parameters directing site map generation process vary from one web site to another. Currently, we simply let the user specify these values. In the future we plan to investigate the ways of selecting them automatically, depending on, for example, the number of pages, or the number of cDocs on the web site.

<sup>8</sup> This style of summary evaluation is one of the most commonly used in summarization community. In DUC 2004, 7 questions similar to ours were used to compare generated and ideal summaries. In our case, however, we do not have ideal summaries, so we cannot follow the same approach. In addition, the purpose of our summaries is different from DUC-style summaries. Rather than summarizing all the important points discussed on the pages belonging to the item, we want our summaries give the user an idea of topic, function, or goal of this particular section of the web site, so that they can decide whether to navigate to that section. Thus, we use a single question reflecting that.



**Fig. 4.** Summary generation results using text centroids (left) and keywords (right)

The results show that the quality of the summaries produced by all methods, except the ones using first sentences, is quite high. The best methods generated 86-87% of good summaries (answer *a* or *b*). The method using all pages of an item produced more summaries of very high quality than other methods. However, it also produced larger number of poor summaries. The reason for that is that if there are multiple topics discussed in a particular section of the web site, and this whole section corresponds to a single site map item, then the method tends to generate a summary describing only the most frequently mentioned topic.

Somewhat surprising was the very good performance of the methods using text of the leader page. We think that, to large extent, this is due to the nature of our dataset. On educational web sites people provide good leader pages that give comprehensive description of the contents of their sections. We plan to conduct more experiments to see whether this will remain true for other domains.

Poor performance of the methods using first sentences of pages is mainly due to the poor quality of the first sentences themselves. While in regular text first sentences often provide an overview of the section, on the Web they tend to contain lists of links to other sections of the web site, or header information irrelevant to the main content of the page. We conclude that such methods are not suitable for summary generation on the Web.

Finally, we did not observe a significant difference in quality between summaries generated using text centroids and keywords<sup>9</sup>.

Overall, the experiments showed that our method could generate high quality site maps. We believe that better parameter tuning can improve the results even further. We also plan to conduct a larger scale experiment to verify that these results hold on a larger number of web sites across multiple domains.

<sup>9</sup> We note, however, that the users who saw our results generally preferred the way we present summaries generated using keyword centroids (see Figure 3). Probably, this is due to the fact that these summaries look similar to the snippets from a search engine results page. In fact, we intentionally tried to make the summaries look similar to Google's search results.

## 5 Conclusion and Future Work

In this paper we described a system for automatic construction of descriptive site maps. The system is based on a combination of machine learning and clustering algorithms, which are used for automatic identification of web site sections. This framework provides a natural way for combining multiple structural and content features of web pages, and allows for automatic selection of the optimal number of sections. The sections are then combined to produce a site map, and multi-document summarization techniques are used to generate anchor texts and summaries for the items. Experimental results on a set of educational web sites show that our method can generate high quality site maps. We believe that our work provides a good example of how Web content and structure mining, combined with natural language processing, can be used together to solve an important practical problem.

In the future we plan to extend this work in several directions. First, we will investigate the problem of automatic parameter selection for site map and summary generation. An important aspect of this problem is automatic evaluation of the quality of a summary and overall quality of a site map. Second, we plan to evaluate our system on a larger number of web sites from multiple domains. Finally, we will explore new browsing paradigms resulting from integrating our approach with a search engine.

## Acknowledgments

This work is supported by the National Science Foundation under Grants No. 0227648, 0227656, and 0227888.

## References

1. Anchor Text Optimization. <http://www.seo-gold.com/tutorial/anchor-text-optimization.html>
2. Candan, K.C., Li, W.-S.: Reasoning for Web Document Associations and its Applications in Site Map Construction. *Data and Knowledge Engineering*, Vol. 43, Issue 2, November 2002
3. Dmitriev, P., Lagoze, C., Suchkov, B.: As We May Perceive: Inferring Logical Documents from Hypertext. *16<sup>th</sup> ACM Conference on Hypertext and Hypermedia*, September 2005
4. Eiron, N., McCurley, K. S.: Untangling compound documents on the web. *14<sup>th</sup> ACM Conference on Hypertext and Hypermedia*, August 2003
5. Eiron, N., McCurley, K. S.: Analysis of anchor text for web search. *26<sup>th</sup> ACM SIGIR Conference*, July 2003
6. Li, W.-S., Kolak, O., Vu, Q., Takano, H.: Defining logical domains in a Web Site. *11<sup>th</sup> ACM Conference on Hypertext and Hypermedia*, May 2000
7. OCLC Web Characterization Project. <http://wcp.oclc.org/>
8. O'Neill, E. T., Lavoie, B. F., Bennett, R.: Trends in the Evolution of the Public Web, 1998-2002. *D-Lib Magazine*, Volume 9, Number 4, April 2003
9. Wall, D.: How to Steal to the Top of Google. <http://www.seoachat.com/c/a/Google-Optimization-Help/>