Deriving Link-context from HTML Tag Tree

Gautam Pant
Department of Management Sciences
The University of Iowa
Iowa City, IA 52242
gautam-pant@uiowa.edu

ABSTRACT

HTML anchors are often surrounded by text that seems to describe the destination page appropriately. The text surrounding a link or the *link-context* is used for a variety of tasks associated with Web information retrieval. These tasks can benefit by identifying regularities in the manner in which "good" contexts appear around links. In this paper, we describe a framework for conducting such a study. The framework serves as an evaluation platform for comparing various link-context derivation methods. We apply the framework to a sample of Web pages obtained from more than 10,000 different categories of the ODP. Our focus is on understanding the potential merits of using a Web page's tag tree structure, for deriving link-contexts. We find that good link-context can be associated with tag tree hierarchy. Our results show that climbing up the tag tree when the link-context provided by greater depths is too short can provide better performance than some of the traditional techniques.

General Terms

Algorithms, Experimentation, Measurement

Keywords

Tag tree, DOM, Link-context

1. INTRODUCTION

Link-context is used for many tasks related with Web information retrieval. In most cases, anchor text or text within an arbitrary size window around a link is used to derive the context of a link. While the importance of link-context has been noted in areas such as automatic classification [2] and search engines [3], *focused* or *topical* crawlers (e.g., [9; 6; 12; 1]) have a special reliance on it. Topical crawlers follow the hyperlinked structure of the Web while using the *scent of information* to direct themselves towards topically relevant pages. For deriving (sniffing) the appropriate scent they mine the content of pages that are already fetched to prioritize the fetching of unvisited pages. Unlike search engines that use contextual information to complement the content based retrieval, topical crawlers depend primarily on contextual information (since they score unvisited pages). Moreover, a search engine may be able to use contextual information about a URL (and the corresponding page) from a large number of pages that contain that URL. Such global information is hard to come by for a topical crawler that only fetches several thousand pages. Hence, the crawler must make the best use of the little information that it has about the unvisited pages. While, the study of combined contexts derived from many pages is important, in this paper we will concentrate on the quality of link-contexts derived from a single page.

A link-context derivation method takes in a hyperlink URL and the HTML page in which it appears as inputs. The output of the method is a context, if any, for the URL. Traditional context derivation methods view an HTML page as a flat file containing text interspersed with links. This view is a simple extension of techniques used with ordinary documents for obtaining contexts of citations included within the documents. However, recent methods have tried to use an HTML document's tag tree or Document Object Model (DOM) structure to derive contexts [5; 2]. These methods view an HTML page as a tree with `<html>` as its root and different tags and text forming the tree nodes. Figure 1 shows an example of an HTML page and its tag tree representation. As noted earlier, the use of text under the anchor tag (anchor text) as link-context has existed for some time. Do other tags and their hierarchy provide any additional information about the context of a link? The question forms one of the underlying themes of this paper.

When we use an anchor text as the context of a link, we are using all the text in the sub-tree rooted at the anchor tag as the link-context. We may extend the idea and consider all the text in the sub-tree rooted at any of the ancestors of an anchor tag as the context of the anchor URL. We call the node where the sub-tree is rooted as the *aggregation node*. Note that both the context and the corresponding link are within the same sub-tree. Hence, we may view an aggregation node as one that aggregates a link with potentially helpful context. There are many potential aggregation nodes for a single anchor.

2. RELATED WORK

Since the early days of Web many different applications have tried to derive contexts of links appearing on Web pages. McBryan [11] used the anchor text to index URLs in the WWW Worm. Iwazume *et. al.* [10] guided a crawler using the anchor text along with an ontology. SharkSearch [9], used not only the anchor text but some of the text in its "vicinity" to estimate the benefit of following the corresponding link. In a recent work, Chakrabarti *et. al.* [5] suggested the idea of using *DOM offset*, based on the dis-

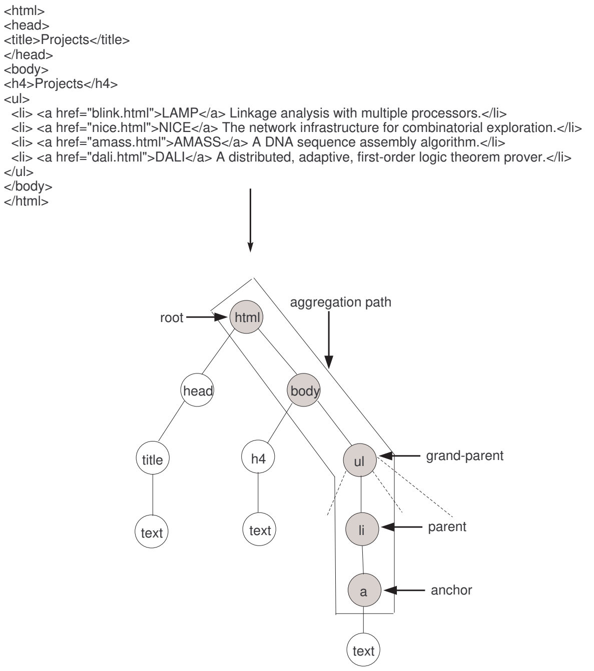

Figure 1: An HTML page and the corresponding tag tree

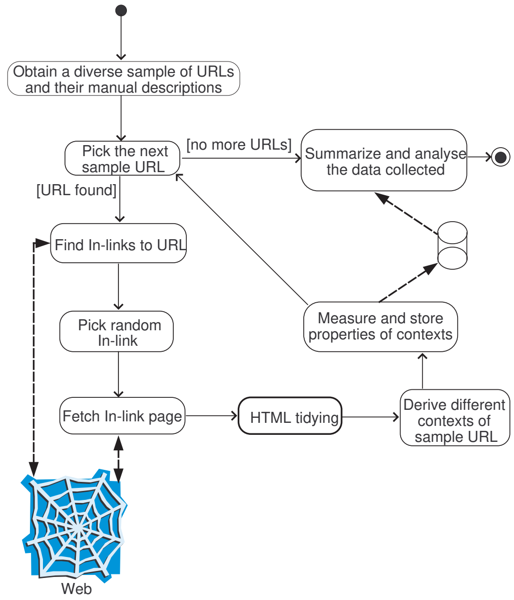

Figure 2: The framework for study

tance of text tokens from an anchor on a tag tree, to score links for crawling. A DOM offset of zero corresponds to all the text tokens in the sub-tree rooted at the anchor tag. The text tokens to the left of the anchor tag on the tag tree are assigned negative DOM offset values. Similarly, positive DOM offset values are assigned to text tokens on the right of the anchor tag. The offset values are assigned in order — the first token on the right or left is assigned a value of ± 1 and so on. The authors trained a classifier in an attempt to identify the optimal DOM offset window to derive the link contexts. They found the offset window to be no larger than 10 (offset values between -5 and $+5$) for most cases. The mechanism for assigning the DOM offset values made no use of the inherent hierarchy in the tag tree. We will later describe link-context derivation methods that make explicit use of the tag tree hierarchy and compare them against some of the traditional techniques.

Attardi *et. al.* [2] proposed a technique that categorized a Web page based on the context of the URL corresponding to the page. In fact, they exploited the HTML structure to derive a sequence of context strings. However, they restricted the analysis to using only a few HTML tags and by obtaining the text within the tags in a selective and ad-hoc manner. We note that their concept of *context path* has similarities to the idea of *aggregation path* described in section *3.3.1*. A context path is a sequence of text strings that are associated with a URL based on their appearance in certain tags and at specific positions in those tags along the hierarchy of the tag tree. The authors also associated a priori semantic meaning with some tags. For example, they argued that the title of a table column or row should be associated with the hyperlinks appearing in the corresponding row or column.

Brin and Page [3] have suggested the use of anchor text to index URLs in their Google search engine. Craswell *et. al.* [7] have shown that using anchor text can provide more effective rankings for site finding tasks. Chakrabarti *et. al.* [4] used a text window of size B bytes around the anchor to obtain the link-context. With an experiment using 5000 Web pages, the authors found that the word "Yahoo" was most likely to occur within 50 bytes of the anchor containing the URL `htpp://www.yahoo.com/`. Davison [8], while using a sample of the DiscoWeb, showed that anchor text (actually it includes words in the vicinity) has high similarity to the destination page as compared to a random page. The author also suggested (based on some results) that enlarging the context of a link by including more words should increase the chance of getting the important terms but at the cost of including more unimportant terms. While the focus of our study is different, we do validate some of the results shown by Davison.

3. FRAMEWORK

3.1 General Methodology

Figure 2 describes the framework in terms of the main steps needed to conduct the study. We first need a sample of URLs that represent a wide variety of topics. In addition, the semantics of the pages corresponding to the URLs must be understood and described by human experts. The manual descriptions will guide us in judging the quality of contexts that are derived for the corresponding URLs. We rely on the manual descriptions because the text content of a Web page may not describe its semantics accurately. Many Web pages are not designed to describe themselves and often contain little text. However, after viewing a page, a human expert can provide us with a good description of what the page is about.

Once we have a sample of Web pages (identified by *sample URLs*) and their manual descriptions, we obtain the in-links for those pages. For each sample URL we use a search engine to find its in-links. We must make sure that the in-link information from the search engine is not stale (i.e. the in-link

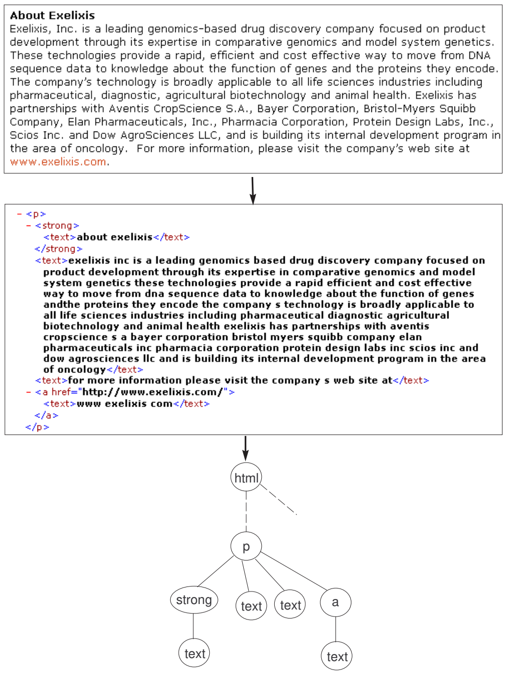

Figure 3: Mapping an HTML snippet (top) to a tag tree (bottom) via conversion of the HTML to a convenient XML format

points to the sample URL). We choose one random in-link for each sample URL. We fetch the page corresponding to the in-link and call it the *in-link page*. The page may provide us with content that describes the sample URL. Before we process the in-link page for deriving contexts, we "tidy" the HTML and convert it into a convenient XML format. After the conversion, we use multiple methods to derive contexts for the sample URL. The manual description of the sample URL helps us in judging different contexts and hence the methods that produced them. In the current study, we use one set of methods that analyze the HTML tag tree and another set that use arbitrary size text windows around the sample URL in `href`. We summarize the performance data obtained from our sample using a number of metrics. The framework allows for evaluating context derivation methods in general. Note that we only fetch the first `20KB` of each in-link page and we parse the tag tree up to a depth of 19 (root of the tree is at depth 0).

3.2 HTML Tag Tree

Many Web pages contain badly written HTML. For example, a start tag may not have an end tag, or the tags may not be properly nested. In many cases, the `<html>` tag or the `<body>` tag is all together missing from the HTML page. The process of converting a "dirty" HTML document into a well-formed one is called tidying an HTML page[1]. It includes insertion of missing tags and reordering of tags in the "dirty" page. Tidying an HTML page is necessary to make sure that we can map the content onto a tree structure with each node having a single parent. Hence, it is an essential precursor to analyzing an HTML page as a tag tree. Further, the text tokens are enclosed between `<text>...</text>` tags which makes sure that all text tokens appear as leaves on the tag tree. While this step is not necessary for mapping an HTML document to a tree structure, it does provide some simplifications for the analysis. Figure 3 shows the process through which an HTML snippet is converted into a convenient XML format and mapped onto a tag tree.

[1]http://www.w3.org/People/Raggett/tidy/

3.3 Context Derivation Methods

We will now describe two link-context derivation techniques. We later evaluate many versions of each and treat each version as a separate context derivation method.

3.3.1 *Context from Aggregation Nodes*

We fetch each in-link page, tidy it up as explained before and map it onto a tag tree structure. Somewhere, on the tree we will find the anchor tag that contains the sample URL. We call that anchor, the *sample anchor*. If there are more than one sample anchors in the in-link page, we consider only the first one for analysis. Next, we treat each node on the path from the root of the tree to the sample anchor as a potential aggregation node (see Figure 1 shaded nodes). Note that a particular context derivation method would identify one aggregation node for each sample URL. The path from the root of the tree to the sample anchor that contains potential aggregation nodes is called the aggregation path (see Figure 1). Once a node on the aggregation path is set to be an aggregation node the following data is collected from it:

- All of the text in the sub-tree rooted at the node is retrieved as the context of the sample URL. The similarity (described later) of the context to the manual description of the sample URL is measured and called the *context similarity*.
- Number of words in the context is counted. Large size contexts may be too "noisy" and burdensome for some systems.

When we decide on a strategy to assign an aggregation node on a given in-link page, it constitutes a context derivation method. Since we have collected the above data for the entire aggregation path for each of the in-links in our sample, we may evaluate many different methods.

For each in-link page we may have an *optimal* aggregation node that provides the highest context similarity for the corresponding sample URL. If there are multiple aggregation nodes with highest similarity, we pick the one closest to the sample anchor as the optimal one.

3.3.2 *Context from Text Window*

We consider the general technique of using arbitrary size windows of text around the sample URL in `href` as a baseline for comparison. Hence, for each sample URL, we use a window of T words around its first appearance on the in-link page as the context. Whenever possible, the window is symmetric with respect to the `href` containing the sample URL. The window may or may not include the entire anchor text. We derive context using multiple values of T starting from 2 and moving up to 100 with increments of 2 words. Each value of T corresponds to a separate context derivation method.

3.4 Performance Metrics

While manual evaluations of context derivation methods are ideal, such evaluation techniques will not scale easily to thousands of sample URLs and several methods applied to each sample. Hence, we depend on automated techniques. Stop words are removed before computing context similarities and stemming [13] is used to normalize the words. The (cosine) similarity between a context c and a description d is computed as:

$$sim(c, d) = \frac{\vec{v_d} \cdot \vec{v_c}}{\| \vec{v_d} \| \cdot \| \vec{v_c} \|}$$

where $\vec{v_c}$ and $\vec{v_d}$ are term frequency based vector representations of the context and the description respectively, $\vec{v_c} \cdot \vec{v_d}$ is the dot (inner) product of the two vectors, and $\| \vec{v} \|$ is the Euclidean norm of the vector $\vec{v}$.

We measure the performance of each of the context derivation methods using the following performance metrics:

Average context similarity: Given a context derivation method $\mathcal{M}$, we measure the context similarity for each of the sample URLs. The average context similarity for $\mathcal{M}$ is then computed as:

$$avg\_context\_sim(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^{i=n} sim(c_i, d_i)$$

where n is the sample size, c_i is the context derived using the method $\mathcal{M}$ for sample URL i, and d_i is the manual description of the sample URL i. The average context similarity is a measure of average quality of contexts obtained using the method.

Zero-similarity frequency: The zero-similarity frequency measures the frequency with which a method obtains a context that has no similarity to the manual description. The frequency is measured as a percentage of the sample size. While zero similarity may not necessarily mean zero information (due to variable word usage), it does mean that the method failed to provide many of the important words that were used by an expert editor to describe a given page. Such failures could have an affect on the recall of a system based on the method. It is possible that a method provides us with high quality contexts for some URLs but provides us with little or no information about many others. The zero-similarity frequency is computed as:

$$zero\_sim\_freq(\mathcal{M}) = \frac{100}{n} \sum_{i=1}^{i=n} \delta(sim(c_i, d_i))$$

where the $\delta(x)$ is 1 if $x = 0$, and is 0 otherwise. Unlike average context similarity, we would like to have a low value of zero-similarity frequency.

Average context size: This measure can be viewed as the potential cost involved in processing or indexing contexts derived from a method. It is simply computed as:

$$avg\_context\_size(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^{i=n} words(c_i)$$

where $words(c)$ counts the number of words in the context c. A desirable property of a context derivation method may be that it produces reasonable size link-contexts.

Note that we associate 95% confidence intervals with all average values.

4. SAMPLE

We obtain our sample URLs from the Open Directory Project[2] (ODP). The ODP contains a large list of categorized Web pages along with their manual descriptions provided by human editors. Due to the lack of commercial bias and easily available content dump, ODP is a suitable and practical source for obtaining our sample. First, we filter out all the categories of the ODP that have no external URLs since we want only those categories from which we can get sample URLs. Next, we remove the categories that appear under the "World", "Regional" or "International" top-level categories or sub-categories. The categories that appear to be an alphabetical category (e.g. Business: Healthcare: Consultancy: A) are also filtered out. These filters are needed to avoid many semantically similar categories as well as pages with non-English content. Once we have a filtered set of ODP categories (nearly a 100,000 in number), we randomly pick 20,000 categories from them. These categories represent a large spectrum of interests on the Web. From each of the 20,000 categories we pick one external URL randomly. The external URL thus obtained is used as a sample URL. We concatenate the title and the description (provided by the editors) of the external URL and use the combined text as the description of the sample URL. Hence, we get a sample of 20,000 URLs along with their manual descriptions from an equal number of ODP categories.

As mentioned earlier in-links to the sample pages are needed to evaluate different context derivation methods. We used the Google Web API[3] to collect the in-links. Using the API we were able to obtain 1-10 in-links for only 14,960 of the 20,000 sample URLs. Out of the in-links returned by the API, we filter out the ones that are from the `dmoz.org` or `directory.google.com` (a popular directory based on ODP) domains. Further, we fetch each page corresponding to the in-links and compare (ignoring case and non-alphanumeric characters) it against the description of the corresponding sample URL. If an in-link page contains the description we filter it out. The filtering process is necessary to avoid ODP pages or their partial or complete mirrors amongst the in-links. Such in-links, if not filtered, will corrupt our conclusions. Finally, we remove in-link pages that do not contain a link to the sample page (a case of stale information from the search engine). We randomly pick one in-link from the remaining in-links and associate it with the sample URL and its description. Table 1 shows a sample URL, its description and a random in-link. Due to the in-link filtering criteria and limited results from the API, our usable sample reduces to 10,549 sample URLs.

5. ANALYSIS

The sample anchor's absolute depth and the number of its ancestors will vary with in-link pages. However, due to the HTML tidying process, we are assured that each sample anchor will have a parent and a grand-parent node. This is because the tidying process inserts `<html>` and `<body>` tags when they are missing from an HTML page. Also, each tag tree will always have `<html>` tag at its root. For some in-link pages, grand-parent and root node will be the same. However, in many other cases the two will be entirely different.

[2]http://dmoz.org

[3]http://www.google.com/apis/

Table 1: Sample instance

Sample URL	*Description*	*In-link*
`http://www.biotechnologyjobs.com/`	biotechnology jobs database of candidate profiles. automatically matches search to most suitable job in the biotech, pharmaceutical, regulatory affairs or medical device industries.	`http://www.agriscape.com/careers/biotechnology/`

Figure 4 shows the performance of four different context derivation methods. The four methods correspond to setting the aggregation node at the anchor (sample anchor), parent, grand-parent and the root node respectively. In addition to the four methods, we also show the performance of a fictional method that would always find the optimal aggregation node. The error bars in this and the following plots show 95% confidence intervals. All the plots with the same performance metric are drawn to same scale. Hence, non-overlapping error bars indicate a result that is significant at $\alpha = 0.05$ significance level. Figure 4(a) shows that the fictional method will have significantly higher average context similarity than any of the other four methods. We also find that the method that uses anchor text gives the highest average context similarity amongst the other four methods. However, the same method leads to information failure for more than 40% of the sample URLs (Figure 4(b)). The average size of the anchor text is found to be 2.85 ± 0.05 words (slightly higher than that reported by Davison [8]). The method that picks context from the parent node, reduces the zero-similarity frequency by more than 11% at the cost of slightly reducing the average context similarity.

The method that sets the aggregation node at the root is using the entire in-link page as a context of the sample URL. As expected such a strategy produces low quality (similarity) contexts with large number of context words (Figure 4(a),(c)). However, it leads to contexts that are far more likely to contain some information about the sample URL (Figure 4(b)). Hence, our metrics show a trade-off which may manifest itself in form of precision and recall in a system that uses the context derivation methods. Davison's [8] results also show a similar trade-off by incrementally enlarging the size of a link-context.

For our baseline methods that use arbitrary size text windows we find that the best average context similarity is obtained when T is about 14 words. Yet, this best average context similarity is significantly worse than that of the method that just sets the aggregation node at sample anchor or its parent. However, the zero-similarity frequency at $T = 14$ is lower (better) than for the methods that uses the anchor or parent as the aggregation node.

We find that for 65% of the sample the optimal aggregation node can be found at the anchor, the parent or the grand-parent nodes. Also, it is most frequently found at the anchor and the frequency progressively decreases as we move up. This suggests that if we were to look for the optimal aggregation node, we should start from the anchor and make our way upwards if needed. Based on this observation, we suggest a simple *tree climbing* algorithm for setting the aggregation node. This algorithm strives for a minimum number of words in a context. We start from the sample anchor and if it does not have enough words (N) under it, we move a level above it in the tree. This process is continued until we have the minimum number (N) of words in our context or we reach the root of the tree. We test the algorithm for various values of N starting from 0 to 20 with increments of 1 word. The algorithm with each value of N can be considered to be a different context derivation method. For $N = 0$, the aggregation node would be trivially set to the sample anchor. Figure 5 shows the performance for various values of N. We find that the average context similarity with $N = 2, 3, 4$ is significantly higher than any of the methods seen before (*cf.* Figure 4(a) and Figure 5(a)). The best average context similarity is seen at $N = 2$ with average context size of about 53 words and zero-similarity frequency of 20% (half of the method using the anchor text). However, even for $N = 2$ the algorithm's average context similarity is much lower than that of the fictional method that finds the optimal aggregation node. Hence, there is large space for improvement that may come through more sophisticated data mining techniques.

6. CONCLUSION

We introduced a framework to study link-context derivation methods. The framework included a number of metrics to understand the utility of a given link-context. Using the framework, we evaluated a number of methods based on HTML tag tree and arbitrary size text windows. The importance of anchor text as link-context was reaffirmed. However, we also noticed its failure in providing information for large number of sample URLs. The parent of an anchor node, appeared to provide a good balance between the lack of information in the case of anchor node and lack of quality in case of root node. We observed that the optimal aggregation node is likely to be found at depths close to the sample anchor. A simple algorithm that utilizes the tag tree hierarchy and the above observation, provided us with higher quality contexts than any of the other methods considered.

We envisage a number of extensions to the current work. Large search engines have many pages that have several in-links. It is worth studying the performance of context derivation methods when they combine contexts from different source pages. In fact, our evaluation framework provides for such an analysis.

7. ACKNOWLEDGMENTS

The author would like to thank Filippo Menczer, Padmini Srinivasan, Alberto Segre and Robin McEntire for their valuable support and suggestions. Special thanks to Nick Street for providing the hardware that was used to conduct the experiments.

8. REFERENCES

[1] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent crawling on the World Wide Web with arbitrary predicates. In *WWW10*, Hong Kong, May 2001.

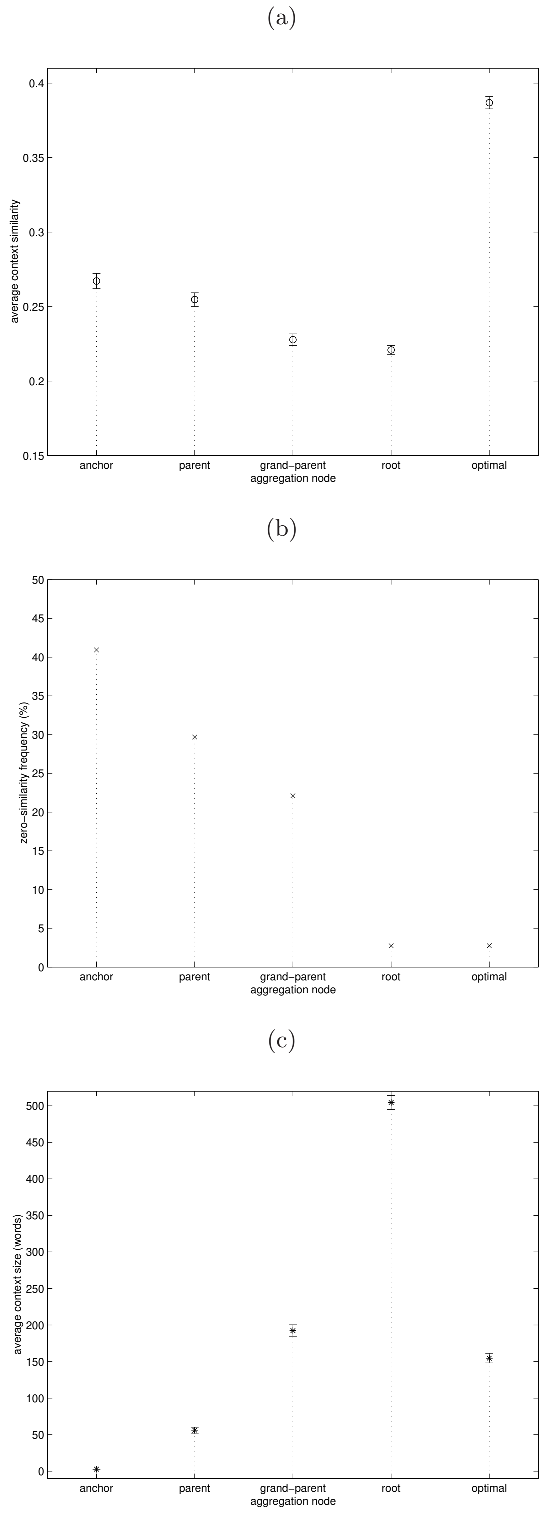

Figure 4: Performance of aggregation node based methods: (a) average context similarity (b) zero-similarity frequency (c) average context size

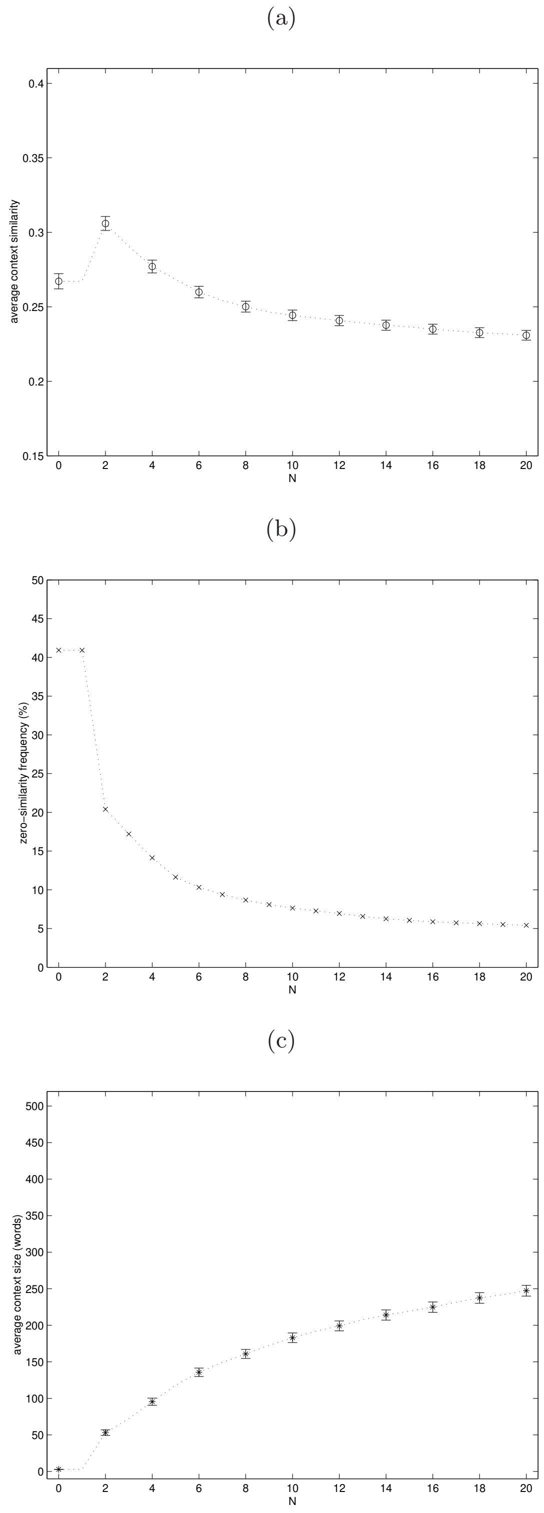

Figure 5: Performance of the tree climbing algorithm: (a) average context similarity (b) zero-similarity frequency (c) average context size

[2] G. Attardi, A. Gullí, and F. Sebastiani. Automatic Web page categorization by link and context analysis. In *Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence*, 1999.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *WWW7*, 1998.

[5] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW2002*, Hawaii, May 2002.

[6] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[7] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *Proc. 24th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[8] B. Davison. Topical locality in the web. In *Proc. 23rd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2000.

[9] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm — An application: Tailored Web site mapping. In *WWW7*, 1998.

[10] M. Iwazume, K. Shirakami, K. Hatadani, H. Takeda, and T. Nishida. Iica: An ontology-based internet navigation system. In *AAAI-96 Workshop on Internet Based Information Systems*, 1996.

[11] O. McBryan. Genvl and wwww: Tools for taming the Web. In *Proc. 1st International World Wide Web Conference*, 1994.

[12] F. Menczer and R. K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. *Machine Learning*, 39(2–3):203–242, 2000.

[13] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.