# EventSeer: Testing Different Approaches to Topical Crawling for Call for Paper Announcements

Knut Eivind Brennhaug

June 30, 2005

# Contents

# Chapter 1

# Introduction

## 1.1 Introduction

The amount of information on the World Wide Web is vast and rapidly increasing. Various tools have been developed to help the user to navigate on the Web and filter out information that is not useful. Search engines provide indexing of crawled pages, but building and maintaining such a library is a time consuming task. If the aim is just to crawl pages about specific topics, performing exhaustive crawls is mostly a waste of time and resources. A topical crawler deal with this problem by focusing the crawl on regions where it is likely to find pages relevant to its topic.

## 1.2 Motivation

Eventseer is an easily navigable search engine that provides access to future conferences in the computer science domain. Today, it automatically determines whether a document is a call for paper announcement, and extracts information such as title, important deadlines, organizing and program committees and important topics. The sources of the documents are mainly different mailing lists. Many call for paper announcements are posted on the Web, and a lot of these are not captured by Eventseer.

## 1.3 Goal

The goal of this master thesis is to explore and test out different topical crawling approaches in order to build a topical crawler that can feed Eventseer with new call for paper announcements from the Web.

# Chapter 2

# Background

## 2.1 Topical Crawling

Topical crawling, or focused crawling, differs from standard exhaustive crawling in the way that it narrows the search down to regions on the Web which are likely to contain relevant pages. An exhaustive crawler just downloads the pages it visits, index them, and follow the hyperlinks without further analysis of the page contents. A focused crawler limits its search to regions of the web that are likely to lead to pages of interest. It uses measures to determine the goodness of a page, and skips further crawling of pages that seem to be fruitless. [CvdBD99] lists the following measures: Site rating, linkage, community behaviour, semi-supervised learning and lifetime.

According to [LMJ04], the Web is growing exponentially, and crawling the whole Web is an infeasible task. In 1999, the largest crawls were covering only about 30-40 percent of the Web, and took weeks to a month to complete [CvdBD99]. In 2005, Google has declared to index 8 billion pages, and tests performed in [GS05] concludes that Google covers 76.1 % of the indexable web.

When only a very small part of the information gathered from a crawl is relevant, one may question whether an exhaustive crawl is worth the effort for others than the large search engines like Google, AlltheWeb and AltaVista. If one lacks the necessary hardware resources to perform exhaustive crawling, the solution is to use a focused crawler that crawls only small parts of the web.

## 2.2 Eventseer

The purpose of the Eventseer project [BS04] is to ease the task of finding relevant call for papers. Conferences are presented in numerous media, ; using mailing lists or the World Wide Web. The different mailing lists have limited range, so several mailing lists are needed to keep the user updated about the events. Typically, only a limited amount of the call for papers are of interest. These circumstances make it a time consuming task for the user to find relevant call for papers.

The purpose of the Eventseer project is to ease the task of finding relevant call for papers. This task is done in two main steps; the search for call for papers, and information extraction to help the user to determine quickly whether the conference is of interest.

In the information extraction step, entities such as important dates, program committee members and important topics are extracted.

The search step includes the harvest of data. Initially, a set of mailing lists are manually determined to be relevant. E-mails are automatically downloaded from the mailing lists, and a classificator is used to filter out e-mails that are not call for paper announcements. According to [BS04], the best classificator was SVM with a vocabulary of 40 n-grams, that achieved an accuracy of 98,7 %. The goal of this master thesis is to develop a topical crawler that can crawl the web for more call for papers.

## 2.3 Related Work

### 2.3.1 Focused Crawling: A New Approach to Topic-Specific Resource Discovery

The concept of focused crawling is introduced in [CvdBD99]. The focused crawler uses a taxonomy of common web topics, and lets the user specify which topics that are of interest. This is used to guide the crawler, which operates with two measures to calculate the goodness of a page; relevance and popularity. Relevance is calculated by analyzing the content, while popularity is calculated by analyzing link structure, a page that is not necessarily relevant, may link to other pages that are.

### 2.3.2 Focused Crawling Using Context Graphs

A focused crawler that uses context graphs is described in [DCL+00]. It addresses a common problem of focused crawling; that a lot of off-topic

pages may lead to a desired page, but that they are discarded from the crawl because they are not considered relevant. The idea of context graphs is to create a model of pages closely linked to a target page, in order to capture discarded crawling paths. The context-focused crawler is built through the following steps:

- build context graphs from a set of seed pages

- use classifiers to learn the context

- let the classifier guide the focused crawler

In the first step, backward crawling is performed on a set of seed pages. Google and AltaVista are queried to find pages linking to them. The procedure is then repeated on the resulting pages. This process is iterated in a user-specified number of steps. The seed pages are assigned to layer 0, pages linking to a seed page are assigned to layer 1, and pages with distance $i$ from a seed page are assigned to layer $i$. Each layer represents an expected distance to a target document.

In the second step, classifiers for the different layers are built. A modified TF-IDF representation is computed for each of the pages, except in cases where the number of pages in a layer is very large, in that case a set of pages from the layer is sampled. The TF-IDF vectors are used to feed the learners. The implementation uses a modification of the Naive Bayes Classifier.

In the last step, the classifiers are used to guide the crawler by assigning a retrieved page to a layer. Pages not assigned to any of the layers are discarded from the crawl.

The result was an improved efficiency compared to a traditional focused crawler.

### 2.3.3 DEADLINER: Building a New Niche Search Engine

DEADLINER [KGC+00] is a search engine that catalogs conference and workshop announcements, and extracts information like important dates, topics and program committees. The DEADLINER architecture forms the basis for Eventseer, and is shown in Figure 2.1, where the crawling is performed in the input stages. The focused crawler that is used, is the one described in [DCL+00].

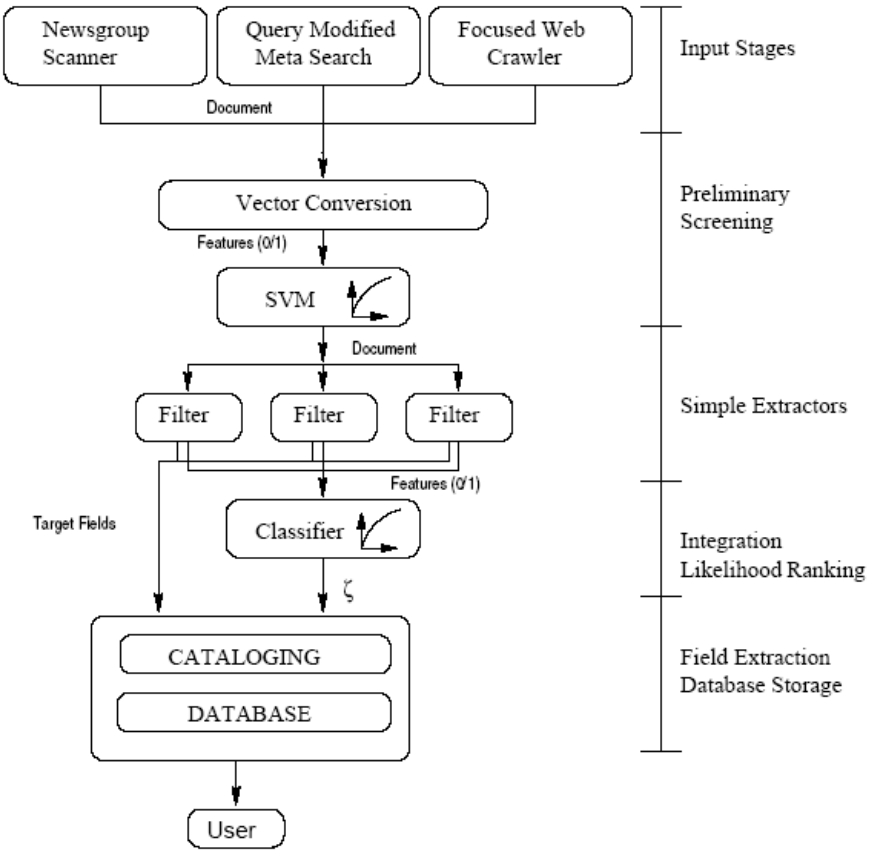


Figure 2.1: DEADLINER Architecture

### 2.3.4 Panorama: Extending Digital Libraries with Topical Crawlers

Panorama is an engine that applies topical crawling to provide additional knowledge and understanding for given research papers. It uses information extracted from a set of documents to query a search engine (Google), and train a classifier to guide the topical crawler. Cluster analysis is performed on the crawled pages.

### 2.3.5 SmartCrawl: A New Strategy for the Exploration of the Hidden Web

SmartCrawl [dCFS04] is a search engine that explores the hidden Web, which is not visible to regular crawlers. SmartCrawl differs from other search engines in that it makes attempts to fill out forms in order to get access to the hidden contents.

### 2.3.6 Probalistic Models for Focused Web Crawling

In [LMJ04], Hidden Markov Models and Conditional Random Fields are used to learn optimal crawling paths that can guide a focused crawler.

# Chapter 3

# Materials and Methods

## 3.1 Materials

### 3.1.1 Programming Languages and Tools

#### Python 2.3.4

Python (www.python.org) is a high-level dynamic programming language created in 1990. It is interpreted, supports object-orientation and has a compact and simple syntax. Writing an application in Python requires significantly less lines of code than if it was to be written in a compiled language like Java or C++, and testing is correspondingly made easier. This makes it suitable for fast development of prototypes where specific implementation details are of less interest. A weakness of Python is its run-time speed, so performance-critical modules may be written in C/C++ and embedded in Python.

In addition to the above mentioned issues, the reasons to choose Python for this project was that it has a built-in module for text processing (regular expressions module), and that a lot of external modules are available, where NLTK and the Google Web API are of particular interest for this project.

#### NLTK - Natural Language Toolkit

The Natural Language Toolkit, NLTK(nltk.sourceforge.net), is a collection of python libraries for natural language processing. While it is mainly intended as a teaching instrument, it can also be used for prototype development and research. Among various items, the tokenizer and the Porter stemmer will be used in this project.

### Porter Stemming Algorithm

Words with common stems often have similar meanings, and therefore classification of documents will be easier if the suffixes of such words are stripped, so that they all become one word. The Porter stemming algorithm [Por80] is an algorithm that perform suffix stripping, and reduces the size of the vocabulary significantly.

### TF-IDF Algorithm

TF-IDF representation (Term Frequency Inverse Document Frequency) is a way of describing a document with a vector, where the vector elements correspond to the weighted frequency of particular phrases in a document. The weighted frequency of a phrase decreases the more often it occurs in the reference corpus. The TF-IDF score *v(w)* of a phrase $w$ is calculated using the following formula:

$$v(w) = \frac{f^d(w)}{f^d_{max}} log \frac{N}{f(w)}$$

where $f^d(w)$ is the number of occurrences of $w$ in a document $d$, $f^d_{max}$ is the maximum number of occurrences of a phrase in document $d$, $N$ is the number of documents in the reference corpus and $f(w)$ is the number of documents in the corpus where the phrase $f(w)$ occurs at least once.

In this project, a reduced TF-IDF score is computed, where only the forty highest scoring components are included in the vector. This is done to increase the speed of the classifiers. A TF-IDF representation is built through the following steps:

1. Remove tags and content between tags that is not shown as plain text
2. Remove stop-words such as "and", "by", "or" and "at".
3. Stem the page using the Porter stemmer algorithm
4. Build a reduced TF-IDF representation using the vocabulary built from the 20 newsgroups dataset

### Support Vector Machines

The original support vector machine (SVM) algorithm is described in [Vap95]. It separates binary classes by placing an optimal hyperplane between them,

and then maximizing the margin to each class. If the examples are not linearly separable, the problem is transformed to a higher-dimensional space where a linear separation is possible. The transformation is obtained by the use of non-linear kernel functions that replace the inner product. Kernels can for example be polynomial or radial-basis functions.

SVM-Light (http://svmlight.joachims.org/) by Joachims is an implementation of Support Vector Machines in C, that solves classification, regression and ranking problems. In this project, it is used for classification of web pages.

### 3.1.2 Datasets

#### 20 Newsgroups Dataset

The 20 Newsgroups Dataset is a corpus of 20,000 messages collected from 20 different newsgroups, where 1,000 messages were sampled from each of the groups. From these messages, a reference vocabulary was built to be used for the TF-IDF algorithm. For each word in the vocabulary, the number of occurrences in the corpus was counted.

#### Training and Test Sets

Call for paper announcements from five different disciplines (informatics, mathematics, physics, chemistry, sociology) were collected. For each discipline, 40 call for paper announcements were collected. The informatics CFPs were collected from Eventseer, while the rest was collected querying Google.

#### Stop-words

There are frequently used words that are so common, that they are rather useless for retrieving documents. Examples are "by", "and", "at" and "or". These are called stop-words, and are ignored by search engines. When preprocessing a page, stop-words are removed to ease the classification. A list of stop words was downloaded from Search Engine World (http://www.searchengineworld.com/spy/stopwords.htm).

## 3.2 Strategies

#### Backward Crawling

Search engines like Google and AltaVista have functions for finding pages that links to a specified page. An initial assumption was that pages linking

to a call for paper announcement also are likely to link to other call for paper announcements (Figure 3.1).

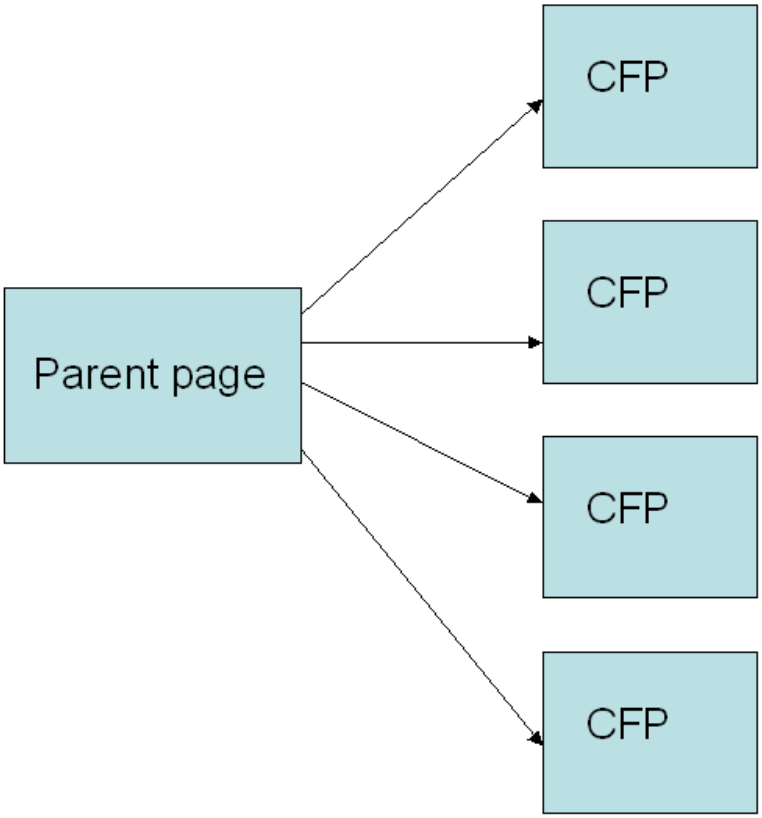


Figure 3.1: Pages linking to a call for paper announcement are also likely to link to other call for paper announcements.

The Python Google API provides methods for querying Google and accessing Google cached pages. A license key is needed, and there is a limit of 1,000 requests per day for each license key. AltaVista search is performed by direct manipulation of the URLs (Figure 3.2) and use of Python libraries for Internet protocols. Manually testing suggests that it is usually AltaVista that provides the best search results.

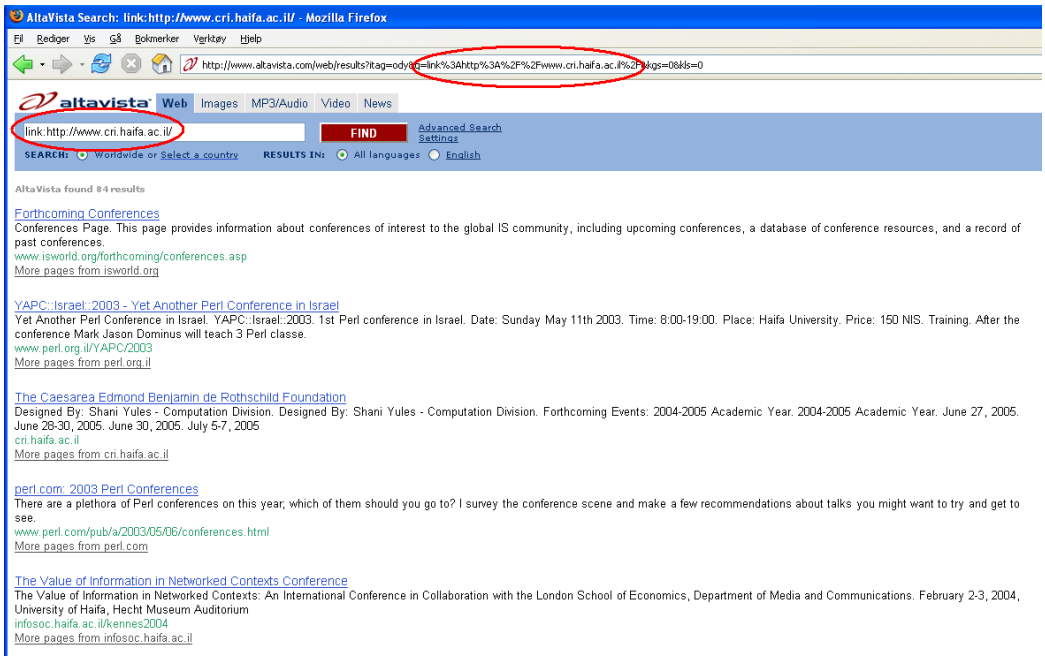


Figure 3.2: AltaVista search by direct manipulation of URL.

### Building Context Graphs

The context focused crawling approach was tested with call for paper announcements. Context graphs with 5 layers were constructed for five disciplines: Informatics, mathematics, physics, chemistry and sociology. Each discipline had a seed set of 40 URLs, and AltaVista was used for back crawling.

### Back-and-Forward Crawling

The idea of back-and-forward crawling is to query a search engine to find the in-links for a page, and do forward crawling on the resulting pages. The process is iterated until the harvest reaches a low, or to a user-specified time limit or harvest limit. An issue that has to be considered is whether the focused crawler shall classify parent pages or child pages.

## 3.3 Crawling Using Context Graphs

The context focused crawling approach was tested with call for paper announcements. The process follows the following steps:

1. build context graphs from the seed sets

2. download the pages from the nodes in the context graphs and preprocess them

3. sample a training set and a test set (that do not overlap) from the whole graph and build classifiers for each layer

4. evaluate results

Context graphs with 5 layers were constructed for five disciplines: Informatics, mathematics, physics, chemistry and sociology. Each discipline had a seed set of 40 URLs to conferences within the discipline, and AltaVista was used for back crawling. Backward crawling was performed in four steps for each of the disciplines. For simplicity purposes, duplicate URLs are not removed.

In the second step, the pages corresponding to the URLs are downloaded. Since the sizes of the layers increases, only a sampled set of the layers are selected for the highest layer.At this stage problems like that a page is unavailable or in a format not directly readable from Python can occur. In this case, the following is done:

- unavailable pages are discarded

- pages in pdf-format are accessed using Google API to access cached versions

- pages in doc-format are discarded

The preprocessing involves the following steps:

1. remove tags and content between tags that is not shown as plain text

2. remove stop-words such as "and", "by", "or" and "at".

3. stem the page using the Porter stemmer algorithm

4. build a reduced TF-IDF representation using the vocabulary built from the 20 newsgroups dataset

To preserve high speed of the classifiers, only the forty highest scoring components are included in the reduced TF-IDF representation.

In the third step, a training and a test set is sampled from the context graph for each discipline. The whole set is sampled at once, and then split into training and test set, so that overlap is avoided. Support Vector Machines are used to build the classifier

Finally, the accuracy of the classifier is tested on the test set. If the accuracy is above a predefined threshold, it is considered good enough to guide the crawler.

## 3.4 Back-and-Forward Crawling

Back-and-forward crawling differs from the context focused crawling in that the crawler only goes back one step, and then goes forward again. In general, back-and-forward crawling is done by iterating the following loop:

1. search for pages linking to a page

2. determine the goodness of each of the pages retrieved

3. go back to the first step for pages classified as good, discard pages classified as not good

There are mainly two ways to determine whether a parent page is good. One can assign it to a predefined class, or one can sample some of the hyperlinks and determine whether they are interesting targets.

Google and AltaVista were queried for pages linking to two large conferences; The 14th International World Wide Web Conference (www2005.org), and 31st Conference on Very Large Databases (www.vldb2005.org). 50 pages from each of the queries were manually assigned a class. After human inspections of the results, the conclusion is that most pages fit into one of the following classes:

- **List of Call for Paper Announcements** Finding a list of CFP announcements (Figure 3.3) is common, and is a very good page to continue crawling with.

- **Institution or Topic Page** This class includes institution pages, e.g. the homepage of an university, and pages dedicated to a specific topic. Pages of this class often include a list of conferences, and may be hard to distinguish from the previous class, even for a human observer. Such pages are usually less clean than pages of the previous category(see Figure 3.4).

- **Conference Page** This class includes homepages of conferences (see Figure 3.5). Information could be extracted, but a page of this class is not very likely to lead to other CFPs. However, it is a hot subject for backward crawling.

- **Mail Announcement** A mail announcement is usually a part of a mail archive, and is likely to lead to other CFPs, and even lists of CFP announcements.

- **Homepage** Quite often a homepage of a program committee member or someone who plans to visit a conference is retrieved. A homepage also contains a list of conferences the owner of the homepage has been to, and conferences that he or she plans to go to. Usually, most links point to off-topic pages, and thus a homepage might be discarded from the crawl.

- **Weblog** A retrieved page might also be a weblog (see Figure 3.6). There is typically an entry that refers to a conference, or perhaps lists some conferences, while the rest of the entries are unrelated.


Figure 3.3: Example of a CFP list.


Figure 3.4: Example of a topic page.


Figure 3.5: Example of a conference page.

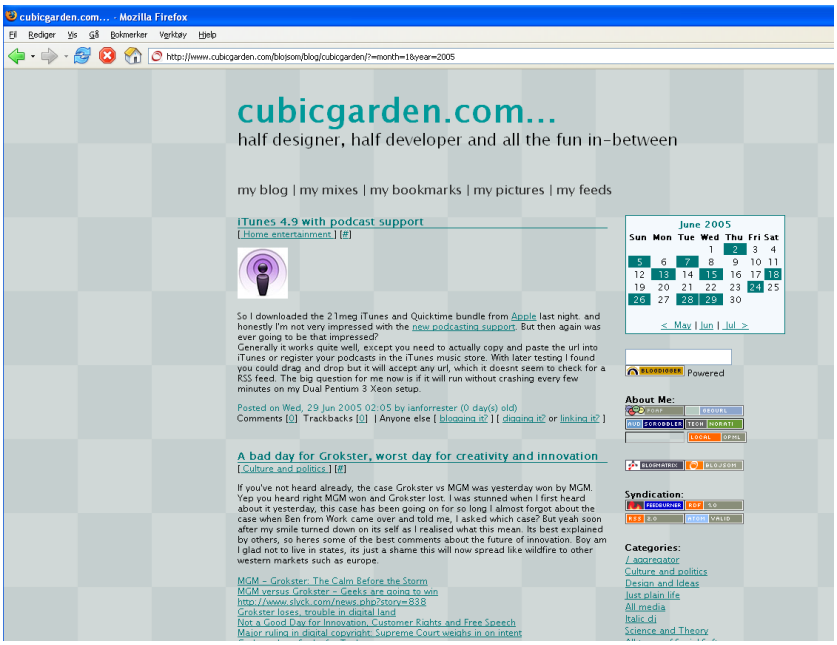


Figure 3.6: Example of a weblog page.

The other way is to classify pages to be CFPs or non-CFPs. This could be done by sampling a small set of the hyperlinks of a parent page. The pages corresponding to the links are then downloaded and classified to be CFPs or non-CFPs. If at least one of these pages are classified as a CFP, all the hyperlinks of the parent page are checked, if not, it is discarded from the crawl. In [BS04], an SVM classifier using feature vectors built from uni-, bi- and trigrams had an accuracy of 98,7 % when classifying e-mails, but classifying web pages is a more challenging task. In most cases, a web page is less structured than an e-mail, and there might be some formatting code that is not removed in the preprocessing step.

In order to build a CFP classifier, the 40 seed URLs for informatics that were used to build a context graph were selected as the positive examples of the training set. 98 negative examples were provided by querying Google for computer science related pages that are not CFPs, job announcements, CFPs from other disciplines than computer science, and some pages neither related to computer science or conferences. The test set contained 79 URLs, where 39 of them were positive examples, and where the negative examples were collected in the same way as the training examples were collected.

The best solution is to combine the two approaches, since what is best to do with a parent page, is dependent of the class it belongs to, if it is

list of CFPs one would like to crawl forward, while one would like to crawl backward if it is a conference page.

# Chapter 4

# Results

This chapter shows the results for the strategies described in chapter 3, the context graphs method and the backward-forward crawling method. The results are followed by a brief discussion.

## 4.1 Context Graphs

### 4.1.1 Layer Sizes

| Layer | Informatics | Mathematics | Physics | Chemistry | Sociology |
|---|---|---|---|---|---|
| 0 | 2 KB | 2 KB | 2 KB | 2 KB | 3 KB |
| 1 | 13 KB | 9 KB | 4 KB | 2 KB | 4 KB |
| 2 | 64 KB | 35 KB | 18 KB | 8 KB | 10 KB |
| 3 | 305 KB | 143 KB | 57 KB | 31 KB | 33 KB |
| 4 | 1,504 KB | 307 KB | 271 KB | 155 KB | 133 KB |

Table 4.1: Sizes of the layers of the context graph for the different disciplines.

As can be seen in Table 4.1, the sizes of the layers seem to grow exponentially, and the context graph for the informatics discipline is growing fastest. A reason for this is that this set entirely consists of fresh URLs from Eventseer, while the other seed sets contain a mixture of new and older CFP pages. The older a CFP page is, the more likely it is that links to it have been removed.

### 4.1.2 Classifications of Layers

The classifier was not at all able to assign a page to a layer. The recall values were very low, and in some cases even zero.

### 4.1.3 Discussion of the Results

The results obtained from the context graphs are clearly unfit for further processing. Possible reasons for the poor results may be the following:

- **Too Small Datasets for the Seed Layer and Layer 1.** The sizes of each of the seeds were 40 pages. Although these are small sets, one would anyway expect that the results were directed in a positive direction despite the classifier being inaccurate
- **Inadequate Preprocessing.** There is room for improvement in the preprocessing stage, but again one would just expect weaker classification accuracies, and not a complete lack of recall.
- **No Typical Representation for the Different Layers.** The results of the backward-forward crawling show that there are different types of pages linking to a CFP announcement. According to Table 4.2, 10 of the 94 parent pages are CFPs themselves, and should consequently be classified to layer 0. After only a few steps of backcrawling, the results tend to be very general. And the results of backcrawling a page, are often pages from the same site. All these issues make it understandable that the classifier fails to build model representations for the different layers.

## 4.2 Backward-Forward Crawling

| Class | Number of Pages |
|---|---|
| CFP list | 24 |
| Inst./Topic page | 13 |
| Conference | 10 |
| Mail announcement | 6 |
| Homepage | 33 |
| Weblog | 18 |

Table 4.2: Distribution of the classes on the training set.

| Class | Number of Pages |
|---|---|
| CFP list | 29 |
| Inst./Topic page | 15 |
| Conference | 17 |
| Mail announcement | 1 |
| Homepage | 33 |
| Weblog | 2 |

Table 4.3: Distribution of the classes on the test set.

| Class | # correct | # incorrect | Accuracy | Precision | Recall |
|---|---|---|---|---|---|
| CFP list | 84 | 13 | 0.866 | 0.767 | 0.793 |
| Inst./Topic page | 82 | 15 | 0.845 | N/A | 0.0 |
| Conference | 80 | 17 | 0.825 | N/A | 0.0 |
| Mail announcement | 96 | 1 | 0.990 | N/A | 0.0 |
| Homepage | 70 | 27 | 0.722 | 0.875 | 0.212 |
| Weblog | 94 | 3 | 0.969 | 0.0 | 0.0 |

Table 4.4: Classification results for the parent classes

| Class | # correct | # incorrect | Accuracy | Precision | Recall |
|---|---|---|---|---|---|
| CFP | 61 | 16 | 0.792 | 0.923 | 0.632 |

Table 4.5: Classification results for CFPs

Table 4.2 shows the distribution of classes in the training set, and Table 4.3 shows the distribution of classes for the test set. One note that the classes Mail announcement and Weblog are rarely represented in the test set, and the classification results for these classes are therefore not of interest. Additionally Table 4.4 says that the classifier was unable to recognize any pages of the classes Institution/Topic page and Conference. The precision/recall values for the CFP lists are reasonable, but not good enough to make the classifier a good guide for the topical crawler. The homepage class has a poor recall value, but quite good precision.

Table 4.5 shows the result for the CFP classifier. It has good precision, but not very good recall. The classifiers from [BS04] were tested on the same sets, and the Orange SVM classifier had a classification accuracy of 84,9 %, which is slightly better than the SVM light classifier. This suggests that use of uni-, bi- and trigrams may be a better option than the TF-IDF algorithm. It also illustrates that classifying web pages is more difficult than classifying e-mails, where Orange SVM achieved an accuracy of 98,7 %.

### 4.2.1 Discussion of the Results

Although the results are far from optimal, the classifier is to some extent able to recognize homepages, and quite good to recognize CFP lists. It is obvious that the test set should be larger, since two classes had too few examples to test the classifier on. Appearantly, text analysis alone does not seem to be sufficient to create a reliable classifier, performing structural analysis of the pages could perhaps aid the classifiers to become more accurate.

# Chapter 5

# Conclusive Remarks

In this paper, different methods for building topical crawlers were investigated.

The context graph method did not at all provide useable results, and the main reason is probably that it is hard to distinguish the different layers from each other. The backward-forward crawling methods looks more promising, but will need improvements in order to make it valuable for practical purposes.

The combination of using a CFP-classifier for target pages and assigning pages linking to target pages to classes is an exciting approach that should be further investigated. By using a larger training and test set, the classifications could be more accurate, and at least be accurate for retrieving CFP lists, which represents the best class for further crawling.

Another area that has potential of improvement is the preprocessing step. A more detailed text analysis could determine what parts of a page that are related to the topic, and what parts that are only weakly related or not related at all. When a good classifier is built, one may advance to the crawling step and extract information from the crawled pages, where especially extraction of important dates could be useful in the crawl.

# Bibliography


[AAGY01] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *World Wide Web*, pages 96–105, 2001.

[BS04] Knut Eivind Brennhaug and Martin Stein. Eventseer: Information extraction of call for paper announcements, 2004.

[CvdBD99] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.

[dCFS04] Augusto de Carvalho Fontes and Fábio Soares Silva. Smartcrawl: a new strategy for the exploration of the hidden web. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 9–15, New York, NY, USA, 2004. ACM Press.

[DCL+00] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[GS05] Antonio Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW (Special interest tracks and posters)*, pages 902–903, 2005.

[HYJS04] Y. L. Hedley, M. Younas, A. James, and M. Sanderson. A two-phase sampling technique for information extraction from hidden web databases. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 1–8, New York, NY, USA, 2004. ACM Press.

[KGC+00] Andries Kruger, C. Lee Giles, Frans Coetzee, Eric Glover, Gary Flake, Steve Lawrence, and Cristian Omlin. DEADLINER: Building a new niche search engine. In *Ninth International Conference on Information and Knowledge Management, CIKM 2000*, pages 272–281, Washington, DC, November 6–11 2000.

[LMJ04] Hongyu Liu, Evangelos E. Milios, and Jeannette Janssen. Probabilistic models for focused web crawling. In *WIDM*, pages 16–22, 2004.

[MPS04] Filippo Menczer, Gautam Pant, and Padmini Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Trans. Inter. Tech.*, 4(4):378–419, 2004.

[PM] G. Pant and F. Menczer. Topical crawling for business intelligence.

[Por80] M.F. Porter. An algorithm for suffix stripping, 1980.

[PTJG] Gautam Pant, Kostas Tsioutsiouliklis, Judy Johnson, and C. Lee Giles. Panorama: Extending digital libraries with topical crawlers.

[Vap95] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.