The Pennsylvania State University
The Graduate School

ALGSEER: AN ARCHITECTURE FOR EXTRACTION, INDEXING AND SEARCH OF ALGORITHMS IN SCIENTIFIC LITERATURE

A Thesis in
Information Sciences and Technology
by
Stephen Carman

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2013

The thesis of Stephen Carman was reviewed and approved* by the following:

C. Lee Giles
David Resse Professor of Information Sciences and Technology
Thesis Advisor

Dinghao Wu
Assistant Professor of Information Sciences and Technology

John Yen
Professor of Information Sciences and Technology

Mahdu Reddy
Associate Professor of Information Sciences and Technology and Graduate Chair

*Signatures are on file in the Graduate School.

Abstract

Algorithms are ubiquitous in the computer science literature. Is is very rare to see a publication in computer science that does not introduce or cite algorithms of some kind or another. It is therefore necessary to extract and index algorithms for search and retrieval. In this thesis we present AlgSeer, a complete architecture for extracting, indexing, and searching for algorithms. We present related work in the areas of citation analysis, document element extraction and speciality search engines that share a similar goal with AlgSeer. We present a complete description of all the pieces that make up the architecture of AlgSeer and we provide in-depth analysis and testing of the each function of the system. We extract algorithms from the set of two million document size CiteSeerX repository and we index and stress test the index of this data. From this data, we extract over 180 thousand algorithms in an XML format totaling 8.3GB. We also provide an analysis of the search showing that our index scales to an extent that far surpasses any realistic prediction of the traffic the system would encounter in any practical scenario with a 600QPM stress test on the index. The query response time never exceeds 5MS.

Table of Contents

List of Figures vi

List of Tables vii

Acknowledgments viii

Chapter 1
Introduction 1

Chapter 2
Related Work 3
2.1 Work in Document Extraction . . . 3
2.1.1 Citation Extraction . . . 4
2.1.2 Other Document Elements . . . 4
2.2 Work in Repositories in Scientific Literature . . . 5
2.3 Scientific Search Engines in Industry . . . 6

Chapter 3
Algorithm Extraction 8
3.1 Extraction . . . 8
3.1.1 Identifying Algorithms . . . 8
3.1.2 Metadata Extraction . . . 9
3.2 Description of the Metadata . . . 9
3.2.1 Extracted Text . . . 10
3.2.2 Extracted Metadata . . . 11
3.2.3 Paper Metadata . . . 11

Chapter 4
Algorithm Indexing Process 13
4.1 Indexing . . . 13

Chapter 5
Algorithm Search Interface 16
5.1 Search . . . 16

Chapter 6
Experiments and Analysis **19**
6.1 CiteSeerX Repository 19
6.2 Performance of Algorithm Extraction 20
6.3 Performance of Algorithm Indexing 20
6.4 Performance of Algorithm Search 22
6.4.1 TF-IDF as a relevancy measure 22
6.4.2 Term Frequency Calculation 23
6.4.3 Inverted Document Frequency 23
6.4.4 TF-IDF Calculation 24
6.5 Solr Stress Testing 24

Chapter 7
Conclusion **27**

Bibliography **29**

List of Figures

3.1 An example of an extracted algorithm. 10

4.1 The Solr XML schema. 14
4.2 The Solr schema presented in its XML format. 15

5.1 The AlgSeer search interface. 17
5.2 The AlgSeer results. 18

6.1 The Term frequency calculation. 22
6.2 Inverted document frequency calculation. 23
6.3 TF-IDF calculation. 24
6.4 Example queries (in Solr format) which are issued to the index. 25
6.5 Pie chart representing the distribution of response times. 26

List of Tables

6.1 Raw system times of running the extraction on 428,837 documents. 20
6.2 Document per minute benchmarks for the extraction process. 20
6.3 Information pertaining to the AlgSeer index and the CiteSeerX repository the information was extracted. 21
6.4 Raw system times of running the indexing function the AlgSeer repository. . . . 21
6.5 Results of the Solr search benchmark using SolrMeter. 25

Acknowledgments

I would like to acknowledge all the intelligent people who helped me to develop and find and succeed on my path. To all the people in my research group, but especially my advisors Lee Giles and Jim Jansen. Also Madian Khabsa, Jian Wu, Wenyi Huang, Suppawong Tuarob, Sumit Bhatia, for all their thoughtful feedback and help with everything. Some people outside my research group, Mo Yu, Yufei Jiang, Bing Zhao, MingYi Zhao, Kayla Booth, and Tristan Endsley for helping to keep me sane. My brother and sister, Paul and Sarah, as well as my girlfriend Kitty, for the endless support and encouragement. And, finally, some really smart people outside Penn State, Andre Calmon and Justin Danielson. I would like to thank everyone for helping me through this challenging time. Your kindness and efforts on my behalf helped me to bring the research presented here to completion.

Dedication

I dedicate my thesis to my parents, Maureen and Paul. Without their love and support none of this would have been possible. Also, the money helped.

Chapter 1

Introduction

An algorithm is a step-by-step descriptions of procedures used for a calculation. They are used in such areas as mathematics and data processing and for some inference and reasoning tasks. Algorithms are everywhere in modern computer science literature. New algorithms as well as surveys and evaluations of current algorithms are being published everyday. The number of algorithms available in the current literature therefore has become staggering. In fact, we have already reached a point at which it is impossible for a human to meaningfully evaluate the algorithms available. Due to the size of this corpus, it is necessary to extract, index, and create a search interface for algorithms in order to help people locate the algorithms best suited for a given purpose.

In this thesis, we introduce AlgSeer, a complete architecture for extraction, indexing and search-

ing for algorithms. AlgSeer is designed to be a free-standing search engine that makes extensive use of free, open-source software that is widely available on the internet. AlgSeer comprises the following functions: an extraction function, based upon previous research pertaining to the extraction of algorithms from scientific documents, an indexing function, which is built on top of Apache Solr and finally a search interface built on top of the Solr Query Language as well as various open source programming languages. When all these pieces are combined, they create a complete search solution for algorithms. In addition, we present an analysis of our extraction, indexing, and searching methods, which are as follows:

- Running the extraction on the entire CiteSeerX repository
- Time taken to extract algorithms and the number of algorithms extracted from the CiteSeerX repository
- Time taken to index 180 thousand documents in Solr
- Stress testing of the search function in AlgSeer

Based on this analysis, we explore the various functions of the system and demonstrate how one can build an entire search solution that scales well and makes use of various open-source software solutions. Finally, we conclude this thesis by summarizing our system together with the analysis results we obtained.

Chapter 2

Related Work

This chapter covers literature and search engines that are related to the presented study. In our discussion, we will focus on two topics. First, we discuss work relating to document element extraction. And then, we will discuss other specialty search engines that seek to accomplish roughly the same thing as AlgSeer, but operate on different document elements. By presenting this background, we will differente AlgSeer from other specialty search engines in the same vein.

2.1 Work in Document Extraction

Significant work has been done in regard to the extraction of specific elements from documents. Specifically, citation extraction is the basis of many major digital libraries such as CiteSeer [1]

and it's successor CiteSeerX[2]. Likewise, citation extraction is at the foundation of industry efforts such as Google Scholar and Microsoft Academic.

2.1.1 Citation Extraction

Citation extraction and analysis has classically been used to measure the impact of research throughout computer science. As a result, the extraction and calculation of various metrics has become an important aspect of digital library research. These metrics include the H-Index for individuals [3] and the impact factor for journals and conferences[4]. Before the age of computers, citation evaluations was performed manually. But with the advent of computers, citation analysis and extraction became has an automatic process[5]. When citations extracted are represented as a graph many improvements results. In addition, citation graphs can be used to supplement search results. Tuarob used a citation graph in supplement algorithm ranking[6] and demonstrated that the graph outperforms the standard TF-IDF in this regard.

2.1.2 Other Document Elements

In addition to citation extraction and analysis, there is considerable work has been done in regards to extraction of different document elements, including Liu et al.'s work on table extraction[7]. Khabsa et al.'s work on acknowledgment extraction should be noted as well[8]. Khabsa et al.'s ap-

proach includes disambiguation in the extraction process and thereby ensure that the appropriate metadata is extracted. Kataria et al. extracted data points and text blocks from two dimensional plots in documents[9]. There is also similar previous work by Lu et al. which seeks to accomplish the same extraction process as Kataria's work above, but through a different method[10].

2.2 Work in Repositories in Scientific Literature

The goal of AlgSeer is to create, an architecture for extracting, indexing and searching for algorithms. The fundamental difference, however: existing search engines operate on different document elements, whereas AlgSeer is specifically designed to locate, extract, and index algorithms.

Teregowdga et al. suggested a system for indexing tables, algorithms, and figures[11]. This system is the basis for a number of the specialty search engine architectures coming out of the CiteSeerX group. However, the difference from this work and what Teregowda proposed is that the search engine was broken out into individual system rather than being integrated into CiteSeerX. Regardless of this Teregowdga et al. provided the groundwork for the indexing process of many of these specialty engines.

TableSeer is a specialty search engine for the extracting, indexing and searching for tables

in scientific literature[7]. In fact, TableSeer is a complete architecture — much like AlgSeer. In addition TableSeer has an accompanying ranking algorithm that is used to rank tables in relation to one another[12]. Some of the ideas for AlgSeer in regard to design build upon TableSeer.

AckSeer, another specialty search engine[8], operates in mostly the same way TableSeer and AlgSeer however, AckSeer, as the name suggests is very specialized because AckSeer operates on the acknowledgments section of given scientific literature. AckSeer also has its own ranking algorithm as well as sophisticated disambiguation functions. In regards to tables and algorithms disambiguation is not relevant as the information extracted does not refer to people or things. For acknowledgments, the large majority of the information is based around people, such that disambiguation is required if correct calculations are to be performed.

2.3 Scientific Search Engines in Industry

It is also worth mentioning the several scientific search engines that have been created by industry. These include Google Scholar[1] Microsoft Academic[2], Wolfram Alpha[3] and Symbolab[4]. Google Scholar as well as Microsoft Academic represent systems very similar to what CiteSeerX strives to be. That is, the overarching goal of all these systems is to offer a complete resource for

[1] http://scholar.google.com
[2] http://academic.research.microsoft.com
[3] http://www.wolframalpha.com
[4] http://www.symbolab.com

academic publications on the web. Though Scholar and Academic also include some level of citation analysis as well as metadata about the papers in their respective repositories. Wolfram Alpha and Symbolab seek to be more specialized than Scholar and Academic. Wolfram Alpha is a semantic search engine that allows more natural queries and in addition provides statistics as well as mathematics in near real time. Symbolab is a mathematics search engine that allows users to search for specific equations or mathematical notations and find more information about a given equation or notation online.

Chapter 3

Algorithm Extraction

3.1 Extraction

The algorithm extraction process comprises two main steps: identifying algorithms and extracting the metadata associated with each of the algorithm identified.

3.1.1 Identifying Algorithms

The method proposed by Bhatia et al. [13] is employed in order to automatically identify algorithms. The method makes two assumptions that an algorithm is represented by a pseudo-code and this code is accompanied by a caption. Based on these assumptions, a set of regular expressions is used to detect what we refer to as "pseudo-code captions."

3.1.2 Metadata Extraction

Once a pseudo-code is identified, the metadata associated with it is also extracted. Such metadata makes the algorithm searchable and includes caption text, reference sentences, overview description, similar algorithms, and other metadata from the paper in which the pseudo-code appears. A pseudo-code reference sentence is a sentence in the document that refers to the pseudo-code. A synopsis is also generated as part of the metadata in order to provide an overview for each pseudo-code extracted. A synopsis of a pseudo-code is generated from the set of its reference sentences, constructed by heuristics and machine-learning techniques [14]. The group of similar algorithms are captured using the algorithm proposed by Tuarob et al. [6], utilizing the semantics from the algorithm co-citation network. An example of an extracted algorithm is below in figure 3.1.

3.2 Description of the Metadata

We will now discuss the various kinds of metadata extracted from scientific documents. First, we will discuss the text extracted from the documents then we will consider the metadata for each algorithm extracted.

```
<doc>
<float name="score">1.0</float>
<str name="algorithm">10.1.1.127.4894_Fig_1</str>
<str name="caption">
Fig 1 Streaming Video using MPEG4 FGS modified greedy algorithm 9
</str>
<str name="checksum">6d4b56f790324765dd72d9ecfa2e6a2a8960021a</str>
<str name="id">10.1.1.127.4894_Fig_1</str>
<int name="ncites">0</int>
<int name="pagenum">1</int>
<str name="paperid">10.1.1.127.4894</str>
<str name="reftext">
Figure 1 illustrates a streaming video system using a FGS codec
</str>
<str name="synopsis">
The bit planes of DCT
trans formed coefficients of these residues
are encoded sequentially to form the FGS enhancement layer
The decoder can decode any truncated segment of the
bitstream of FGS layer corresponding to each frame
The more bits the de coder receives and decodes the
higher the video quality is Figure 1 illustrates a
streaming video system using a FGS codec The decoder can
decode the received truncated bitstream Figure 2 shows
a block diagram of our proposed dis tortion management
protocol to transmit FGS video over multicode CDMA
</str>
<int name="year">2000</int>
</doc>
```

Figure 3.1. An example of an extracted algorithm.

3.2.1 Extracted Text

Three kinds of extracted text are created for each algorithm: reference text, caption text and synopsis text. The reference and caption text are extracted directly from the paper itself. The reference text is the text in which the algorithm is embedded. This text is generated by locating references to the algorithm itself in the text. The caption text is the direct caption text of the algorithm. The synopsis text is generated text based on the references of the algorithm in the

text itself. Synopsis text is based on the work of Bhatia et al.[14] and generates some key pieces of text selected in an effort to explain the algorithm by how it is referenced in the text. These text fields are all indexed in the AlgSeer architecture and are used as the primary fields by the search function of the system.

3.2.2 Extracted Metadata

In addition to the text, various elements of metadata are extracted that describe the algorithm as well as its location and references to it in the text. These elements are id, paper id, and page number. Given the possibility that more than one algorithm can be extracted from a single paper the id is used internally to distinguish between multiple algorithms. The paper id is generally capable of specifying which algorithms or set of algorithms belong to a paper. The page number or page numbers refer to the page(s) in the document on which the algorithm appears. This is a user-friendly feature of the system as it tells users exactly where in a given document the algorithm they are seeking can be found.

3.2.3 Paper Metadata

Finally, paper metadata refers to information about the specific papers from which the algorithms are extracted from. Taken directly from the CiteSeerx database this metadata and is based on the

docID given to each document by CiteSeerX. Given that the papers are taken from the CiteSeerX repository all of the docIDs issued to these papers came with the repository. In addition, the number of times the paper is cited and finally, the year the paper was published. Users use this data to tell how recent the paper is and they can get an idea of it's impact on the field based on the number of times it has been cited in other papers.

Chapter 4

Algorithm Indexing Process

4.1 Indexing

AlgSeer uses Apache Solr for its indexing purposes.[1] The extraction process outputs indexable XML files based on the Solr schema. New papers from the CiteSeerX crawl database are retrieved periodically. The extraction process is run on these papers and an XML file is generated for each paper. These XML files are stored with the PDF files that was used to generate the XML file in the same directory. This is so the XML file can be located easily with it's respective PDF file. These XML files also allow the index to be rebuilt at a later time should it ever become lost. After the XML files are generated and stored they are posted to the Solr index via a RESTful

[1] http://lucene.apache.org/solr/

Field	Datatype	Indexed?
ID	Integer	No
Caption	Text	Yes
RefText	Text	Yes
Synopsis	Text	Yes
PaperID	Integer	No
PageNum	Integer	No
Year	Integer	No
Num of Citations	Integer	No

Figure 4.1. The Solr XML schema.

call. A RESTful call is a simple method of transferring data over an HTTP connection. Once all the new papers have been sent the index is committed and updated. This update is performed in a batch-related process using the post.jar tool provided by Solr. This process is performed once a day to prevent any slow down of services due to the process of updating of the index. However, as Solr is at the enterprise level, it's not expected to happen in the near future. The extraction process removes any and all bad characters that may prevent the XML files from being able to be indexed. Therefore there have not been any problems with using the Solr provided tools to index the newly created XML files. The XML files generated conforms to the internal Solr Schema, which is provided in figure 4.1.

Some aspects of the schema are fairly easily apprehended at a glance (such as id) however others are not. Caption refers to the caption text extracted above or below an extracted algorithm or pseudo-code. Reference text (or reftext in the schema) is text extracted that references the algorithm with in the paper text. Synopsis refers to an overview of the text extracted from

each algorithm. Finally, page number refers to the page that the algorithm appears in the text, whereas the year refers to the year in which the paper was published, and number of citations was the number of times the paper has been cited according to CiteSeerX.

```
<field name="id" type="string" indexed="true" stored="true"/>
<field name="algorithm" type="text_general" indexed="true" stored="true"/>
<field name="caption" type="text_general" indexed="true" stored="true"/>
<field name="reftext" type="text_general" indexed="true" stored="true"/>
<field name="synopsis" type="text_general" indexed="true" stored="true"/>
<field name="paperid" type="string" indexed="false" stored="true"/>
<field name="pagenum" type="int" indexed="false" stored="true"/>
<field name="year" type="int" indexed="false" stored="true"/>
<field name="ncites" type="int" indexed="false" stored="true"/>
<field name="checksum" type="string" indexed="false" stored="true"/>
```

Figure 4.2. The Solr schema presented in its XML format.

There are a couple of important details to take note of in the above text. First, it is important to note that not everything is indexed. This is unnecessary and would add significantly to computational cost when indexing the document. As a result, certain metadata is not indexed. Most of the this metadata is meant as a connection between the index and the CiteseerX database so it's purpose was not meant as something to be searched on in first place. As a result, the most general text fields are indexed because it is the intention to search upon those fields, while most raw data types are not indexed because they exist as supporting information to the algorithms.

Chapter 5

Algorithm Search Interface

5.1 Search

The search function of the engine is available openly on the Internet.[1] This search functionality is an interface to our indexed algorithms. It allows the user to define the text within which they wish to search. The search function also allows certain fields to be boosted (also negatively, deprioritizing them) it also enables the user to search every field with equal boosting. This interface is based on the Lucene query syntax.[2] The query is then submitted to the index. The index uses the TF-IDF in order to compute the score of the given algorithm in relation to the user's query. Whereas previous work [6] uses a citation graph to improve these results, we wanted

[1] http://aspen2.ist.psu.edu/
[2] http://goo.gl/orsVy

a solution that would not require the entire CiteSeerX dataset. In addition, we wanted the user to have more control in regard to refining and changing the query so that he/she would be able to indicate the relative importance of the terms and have more control over which text to search. The scores of each individual algorithm are provided so that users can see how the algorithms relate to one another in reference to the score. Scores are assigned based on their TF-IDF of each algorithm and then the set of documents is returned and presented to the user allowing he/she to view the document in CiteSeerX. The interface is written in PHP and Apache is used to serve the requests. AlgSeer does not maintain its own database, although it will in the future. Currently, all database-related actions are either obtained from index metadata or directly from the CiteSeerX database. The interface itself is fairly fast providing feedback for even large sets of results close to instantaneously. This speed arises from the fact that the query are being encoded and sent using the cURL library. This query then returns a JSON object, which is decoded into the HTML results. This process has no noticeable slowdown in it's use even when returning 50,000+ results for a query.

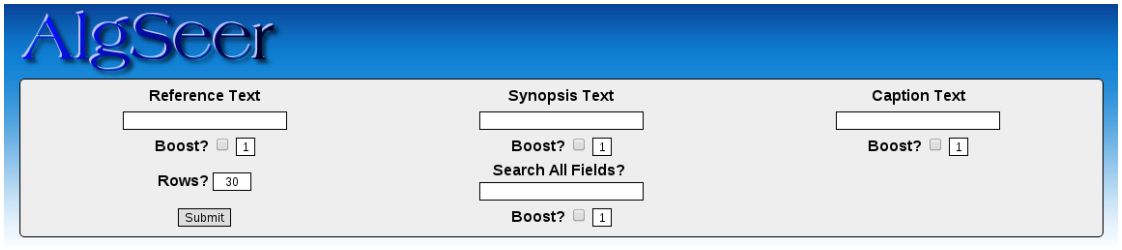

Figure 5.1. The AlgSeer search interface.

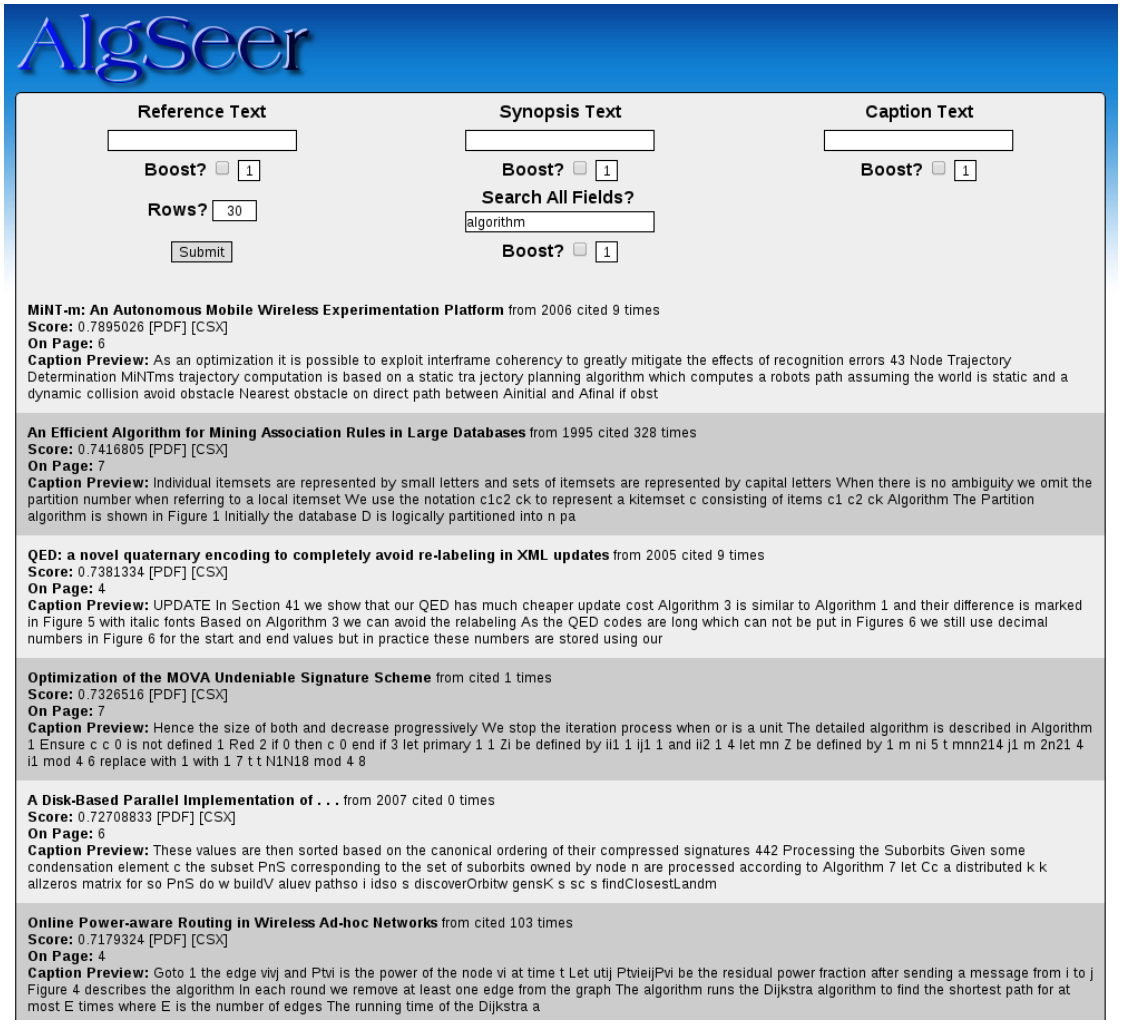

Figure 5.2. The AlgSeer results.

Chapter 6

Experiments and Analysis

In this section, we run experiments and analyze the results on aspects of the system. To start, we will talk about benchmarking of the algorithm extraction process, which was run on the entire CiteSeerX repository. Then we will talk about bench marking of the indexing process on the XML files generated from the algorithm extraction. Finally, we will talk about the benchmarking of the search interface with a stress test.

6.1 CiteSeerX Repository

Most of the data used here is based on a snapshot of the CiteSeerX repository taken in March 2013. We will now detail some information pertaining to the system.

6.2 Performance of Algorithm Extraction

Algorithm extraction is the process whereby algorithms are extracted from PDFs and converted into an indexable XML file. This process is described in detail in Chapter 3, therefore, we will not describe it in detail again here.

All tests for these benchmarks are run on a 2.67GHz Intel Processor with 4 cores and 16GB of ram. The system has approximately 1TB of hard disk space and runs Red Hat 6.4 on the 2.6.32-358 Linux kernel.

Type	Raw time	Hours	Minutes
Real	3647m 36.911s	60	3647
User	2393m 6.854s	39	2393
Sys	675m 18.376s	11.25	675

Table 6.1. Raw system times of running the extraction on 428,837 documents.

Type	Documents per hour	Documents per minute
Real	7147	119
User	10995	183
Sys	38118	635

Table 6.2. Document per minute benchmarks for the extraction process.

6.3 Performance of Algorithm Indexing

The indexing process is integral to the proper functioning of the entire system. Indexing can be expensive computationally. Therefore, it is important to have an indexing system with appropriate trade-offs in regard to time vs. cost. With this in mind, we designed AlgSeer's indexing

function to operate in a batch related process. The batch process was selected in order to minimize the amount of downtime on the system. Additionally, AlgSeer uses Solr, an enterprise-level piece of software for its indexing function. Solr is known to easily scale into the millions of documents — with even meager hardware. We timed Solr's ability to index the entire CiteSeerX repository. This approach is not realistic in practice as we are unlikely to want to index such a vast amount of material, but it gives a good indication of the performance of the index under stress. Table 6.4 is the results of indexing over 2 million documents from the CiteSeerX repository.

CSX Repository	AlgSeer Repository
2.7TB	8.3GB
2104780 Documents	189213 Documents

Table 6.3. Information pertaining to the AlgSeer index and the CiteSeerX repository the information was extracted.

Type	Raw Time	Hours
Real	4913m 56.579s	81
User	3210m 36.488s	53.5
Sys	581m 29.034s	9.6

Table 6.4. Raw system times of running the indexing function the AlgSeer repository.

As we can see from Table 6.4 the indexing is fairly fast. The results show that our system indexed roughly 25,984 documents an hour. And it should be noted, that we predict that our system is capable of greater speed, as these results reflect issues with the argument list using the Solr provided tools to index the XML files. Overall, the Solr index from the AlgSeer repository

was 387MB in size.

6.4 Performance of Algorithm Search

We will now discuss the performance of the searching aspect of AlgSeer. Currently AlgSeer indexes 188,432 algorithms. First, we will discuss how relevancy is calculated in AlgSeer. Then, we will discuss stress-testing of the index to see how it scales.

6.4.1 TF-IDF as a relevancy measure

Currently, AlgSeer uses a standard TF-IDF metric on the text extracted to determine relevancy. This method is fairly standard and constitutes a good starting point in regard to determining relevancy. TF-IDF or term frequency-inverse document frequency is a statistic that reflects how important a term is to a collection of documents. TF-IDF, therefore, provides a very natural ranking to the documents based on the score computed.

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d):w \in d\}}$$

Figure 6.1. The Term frequency calculation.

6.4.2 Term Frequency Calculation

We begin by calculating the normalized term frequency. This is done by taking the ratio of the term's appearance in a document $f(t, d)$ over the term appearing the most times in a document. The goal of determining this ratio is to prevent longer documents from being prioritized over shorter documents, as when using a simple raw term frequency longer documents are favored. This is the case because they have more terms overall and can therefore have larger coverage of the corpus. So, the term frequency is normalized in order to prevent the system bias in favor of longer documents.

$$idf(t, D) = log \frac{|D|}{1+|\{d:t_i \in d\}|}$$

Figure 6.2. Inverted document frequency calculation.

6.4.3 Inverted Document Frequency

Next, the inverted document frequency or IDF is calculated for a term t in a collection of documents, D as seen above in figure 6.2. The IDF is a measure of the rarity of a term in a corpus of documents and it is calculated by taking the logarithm of the ratio of the total number of documents in a corpus by the number of documents in the corpus containing the specific term. The addition of 1 to the denominator is meant to prevent a division by error in cases in which the term does not appear in any of the documents included in the corpus.

$$TF - IDF(t, d, D) = tf(t, d) \times idf(t, D)$$

Figure 6.3. TF-IDF calculation.

6.4.4 TF-IDF Calculation

Finally, we calculate the complete TF-IDF calculation. This is done simply by multiplying the normalized term frequency by the inverted document frequency. This calculation gives a score for each document in the corpus. And, based on this score we can easily rank the documents in relation to the search term.

6.5 Solr Stress Testing

We now look at stress testing the Solr index. The Solr index scales very well; therefore we used a high stress configuration was used. Though the actual AlgSeer does not even see 1QPM on average the Solr index was tested using over 600QPM, or 10 queries a second to the index. Figure 6.4 shows some example queries extracted from query logs and used to test the index. A group of random queries was sampled from a manually created list and used at random to test the scaling ability of the Solr index.

The search all functionality of the query was used most here. This functionality was selected because it puts the most stress upon the index given that it has to search across 3 fields of indexed

```
reftext:queue AND synopsis:queue AND caption:queue
reftext:scheduling AND synopsis:scheduling AND caption:scheduling
reftext:optimization AND synopsis:optimization AND caption:optimization
reftext:reference AND synopsis:reference AND caption:reference
reftext:swarm AND synopsis:swarm AND caption:swarm
reftext:baysein AND synopsis:baysein AND caption:baysein
```

Figure 6.4. Example queries (in Solr format) which are issued to the index.

text and compile the results. Query responses ranged from 50 results to 92,642 results. Now we will look at the results of the stress test itself. This test was run over a period of 16.5 hours and was ran on 4 cores and 16GB of ram. During the time of the test, the CPU never exceeded 16% utilization across all 4 cores and no more than 3GB of ram was allocated at any given time.

Total queries	562568
Total query time	722674ms
Average query time	1.28ms
Queries per minute	600
Queries per second	10
Actual QPM	552
Actual QPS	9.2
Standard deviation	0.92
Total running time	16.5 hours

Table 6.5. Results of the Solr search benchmark using SolrMeter.

In explaining these results, we see that roughly 562 thousand queries were issued to the index over a 16.5 hour period. The average query response time ranged from under 1ms to 5ms. However, 4ms and 5ms are not represented in the Figure 6.5 because they represent a very small fraction of the results in relation to the rest of the data. To include these results would not have been meaningful. We see that overall the average response time was 1.28ms or 0.00128 seconds. Even the slowest query response of 5ms is still well under a second. As far as scaling goes, the

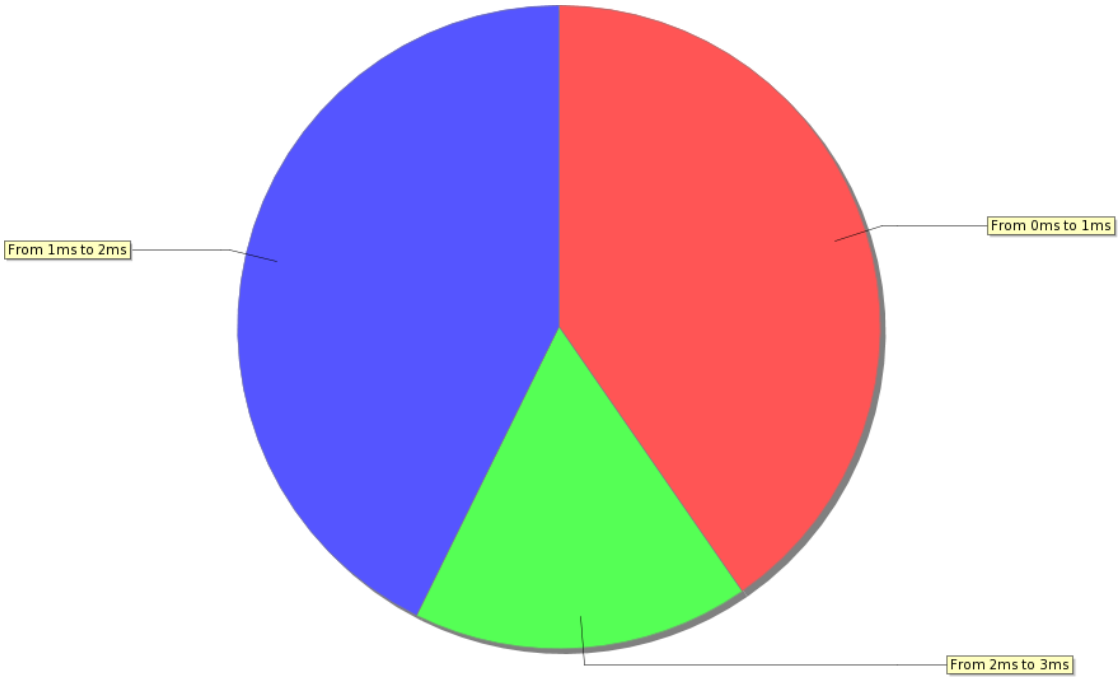

Figure 6.5. Pie chart representing the distribution of response times.

results suggest that the Solr index will scale for a significant amount of time and return results in a timely manner before encountering issues. These statistics correlate with the internally generated Solr statistics, which give an average query time of 1.68ms.

Chapter 7

Conclusion

In this thesis we have presented AlgSeer, a complete search engine for extracting, indexing, and searching for algorithms. We have presented the work related to this project including work in citation extraction and understanding as well as other similar specialty search engines based around different document elements in the scientific literature.

We have explained each individual aspect of the system, the extraction method based upon previous work by the CiteSeerX group, the index function based on the piece of open-source software Solr and the searching interface. We have discussed various benchmarks and the stress testing we subjected various parts of our system to. We ran the extraction process on the entire 2.7TB CiteseerX repository and extracted over 180 thousand algorithms from the repository. We then tested how fast we could index all the XML files extracted from the repository and we

finished by stress testing the Solr index. On this point we showed that it scales well beyond the magnitude of any query that we would expect the system to be called upon to address facing the system in terms of real world traffic.

To finish, we will discuss some of the limitations of the system. AlgSeer does not maintain it's own database. Rather, it uses some metadata to connect with the CiteSeerX database. In the future, it's envisioned that AlgSeer will have it's own database. Also, most of the pieces of the system are disconnected. This means they need to be ran manually at interval levels in the system. We would like to have the finishing of one function in the system begin the next step to give AlgSeer totally automated extraction, indexing and search functions.

Bibliography

[1] Giles, C. L., K. D. Bollacker, and S. Lawrence (1998) "CiteSeer: an automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries - DL '98*, ACM Press, New York, New York, USA, pp. 89–98.

[2] Li, H., I. Councill, W. Lee, and C. Giles (2006) "CiteSeerx: an architecture and web service design for an academic document search engine." in *Proceedings of the 15th international conference on World Wide Web*, pp. 2–3.
URL http://dl.acm.org/citation.cfm?id=1135926

[3] Hirsch, J. E. (2005) "An index to quantify an individual's scientific research output." *Proceedings of the National Academy of Sciences of the United States of America*, **102**(46), pp. 16569–72.

[4] Garfield, E. (2006) "The history and meaning of the journal impact factor." *JAMA : the journal of the American Medical Association*, **295**(1), pp. 90–3.
URL http://www.ncbi.nlm.nih.gov/pubmed/16391221

[5] Lawrence, S., C. L. Giles, and K. Bollacker (1999) "Digital libraries and autonomous citation indexing," *Computer*, (June), pp. 67–71.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=769447

[6] Tuarob, S., P. Mitra, and C. L. Giles (2012) "Improving algorithm search using the algorithm co-citation network," *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries - JCDL '12*, p. 277.
URL http://dl.acm.org/citation.cfm?doid=2232817.2232869

[7] Liu, Y., K. Bai, P. Mitra, and C. L. Giles (2007) "TableSeer : Automatic Table Metadata Extraction and Searching in Digital Libraries Categories and Subject Descriptors," in *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*.

[8] Khabsa, M., P. Treeratpituk, and C. Giles (2012) "AckSeer: a repository and search engine for automatically extracted acknowledgments from digital libraries," *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pp. 185–194.
URL http://dl.acm.org/citation.cfm?id=2232852

[9] Kataria, S., W. Browuer, P. Mitra, and C. Giles (2008) "Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents," in *Proceedings of the 23rd national ...*, pp. 1169–1174.
URL http://www.aaai.org/Papers/AAAI/2008/AAAI08-185.pdf

[10] Lu, X., J. Wang, P. Mitra, and C. Giles (2007) "Automatic Extraction of Data from 2-D Plots in Documents," *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, pp. 188–192.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4378701

[11] Teregowda, P. B., M. Khabsa, and C. L. Giles (2012) "A system for indexing tables, algorithms and figures," *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries - JCDL '12*, p. 343.
URL http://dl.acm.org/citation.cfm?doid=2232817.2232882

[12] Liu, Y., K. Bai, P. Mitra, and C. L. Giles (2007) "TableRank : A Ranking Algorithm for Table Search and Retrieval TableSeer : The Table Search Engine," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*.

[13] Bhatia, S., S. Tuarob, P. Mitra, and C. L. Giles (2011) "An algorithm search engine for software developers," *Proceeding of the 3rd international workshop on Search-driven development: users, infrastructure, tools, and evaluation - SUITE '11*, pp. 13–16.
URL http://portal.acm.org/citation.cfm?doid=1985429.1985433

[14] Bhatia, S. and P. Mitra (2012) "Summarizing figures, tables, and algorithms in scientific publications to augment search results," *ACM Transactions on Information Systems (TOIS)*, **2**(3).
URL http://dl.acm.org/citation.cfm?id=2094075

[15] Bhatia, S., S. Lahiri, and P. Mitra (2009) "Generating synopses for document-element search," *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, p. 2003.
URL http://portal.acm.org/citation.cfm?doid=1645953.1646287

[16] Bhatia, S. and P. Mitra (2009) "Synopses Generation for Specialized Document-Element Search Engines," in *Workshop on Web Search Result Summarization and Presentation, Co-Located with WWW*.
URL http://www.personal.psu.edu/users/s/u/sub194/WSSP2009.pdf

[17] Bhatia, S., P. Mitra, and C. L. Giles (2010) "Finding algorithms in scientific articles," *Proceedings of the 19th international conference on World wide web - WWW '10*, p. 1061.
URL http://portal.acm.org/citation.cfm?doid=1772690.1772804

[18] Salton, G., A. Wong, and C. Yang (1975) "A Vector Space Model for Automatic Indexing," *Communications of the ACM*, **18**(11).
URL http://dl.acm.org/citation.cfm?id=361220

[19] Teregowda, P. and I. Councill (2010) "Seersuite: Developing a scalable and reliable application framework for building digital libraries by crawling the web," *Proceeding WebApps'10 Proceedings of the 2010 USENIX conference on Web application development*.
URL http://dl.acm.org/citation.cfm?id=1863166.1863180

[20] Tuarob, S., L. Pouchard, and N. Noy (2012) "ONEMercury: Towards Automatic Annotation of Environmental Science Metadata," *AGU Fall Meeting*, pp. 1–12.
URL http://www.personal.psu.edu/szt5115/publications/LISC2012.pdf