

The Pennsylvania State University
The Graduate School
College of Engineering

TOWARDS BETTER ACCESSIBILITY OF SCHOLARLY DATA

A Dissertation in
Computer Science and Engineering
by
Madian Khabsa

© 2015 Madian Khabsa

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2015

The dissertation of Madian Khabsa was reviewed and approved* by the following:

C. Lee Giles

David Reese Professor of Information Sciences and Technology

Professor of Computer Science and Engineering

Dissertation Advisor, Co-Chair of Committee

Daniel Kifer

Associate Professor of Computer Science and Engineering

Co-Chair of Committee

Wang-Chien Lee

Associate Professor of Computer Science and Engineering

Murali Haran

Associate Professor of Statistics

Lee Coraor

Associate Professor of Computer Science and Engineering

Graduate Program Chair

*Signatures are on file in the Graduate School.

Abstract

This dissertation focuses on studying the accessibility of scholarly data and introduces methods to facilitate researcher's access to numerous types of scholarly data. Four problems are investigated in the hunt for this mission. At first, an estimate of the total number of scholarly documents on the web is provided using capture/recapture methods. The percentage of publicly available scholarly documents, of the total, is further estimated and provided for each scientific field. It is found that at least 114 million scholarly documents exist on the web as of January 2013.

Secondly, methods for automatically extracting content from scholarly documents are visited in the chemistry domains where formulae and chemical names are extracted automatically. Two methods of extracting these entities are introduced. The first approach introduces a probabilistic framework that combines the output of an ensemble of extractors that can be used as-is. Joining the extractors in a probabilistic approach is shown to achieve higher F score than each individual extractor. The second approach creates a single Conditional Random Fields extractor with novel feature set that utilizes Soundex codes and word embeddings to achieve high accuracy.

Thirdly, the interactions between users of a digital library and the system are analyzed through large scale access log mining. The goal is to uncover user behavior when interacting with a scholarly digital library and search engine. Three years of access logs from CiteSeerX are analyzed, where user search, and download behavior are modeled. It is found that the number of downloads a given paper receives over a period of time can be modeled as power law distribution. Similarly, the length of user sessions exhibits a power law distribution. Furthermore, it was found that the top 1% most downloaded papers in a given month from a digital library are responsible for 10-20% of the overall traffic. In addition, these most downloaded papers exhibit a repetitive pattern where they remain highly accessed in the following months. Therefore, a logistic regression model is introduced to predict the highly accessed papers for a given month using historical data.

Finally, academic search is studied. Search is an essential way for the discov-

erability of scholarly documents. First, academic user query intent is classified in to informational and navigational. We define the multiple facets of navigational queries and introduce a method for identifying them using boosted trees. Later, the clickthrough logs of academic search are studied, and found to have moderate correlation with the relevance of a manually judged dataset, indicating that raw click counts may be used to learn ranking functions. Finally, we use a learning to rank model to devise a ranking function for academic search. The usefulness of the features within the learning to rank model are examined, where it was found that article full text is one of the key signals for predicting relevance.

Table of Contents

List of Figures	viii
List of Tables	x
Acknowledgments	xii
Chapter 1	
Introduction	1
1.1 Estimate of the Number of Scholarly Documents on the Web	3
1.2 Chemical Entity Extraction	4
1.3 Log Analysis of CiteSeerX	5
1.4 Academic Documents Search	5
Chapter 2	
The Number of Scholarly Documents on the Public Web	6
2.1 Introduction	6
2.2 Estimating the Number of Scholarly Documents on the Web	8
2.3 Field Level Analysis	13
2.4 Conclusion	18
2.5 Additional Estimates	18
Chapter 3	
Chemical Entity Extraction	21
3.1 Introduction and Related Work	22
3.2 Methods	25
3.2.1 Ensemble extraction - pre competition	25
3.2.1.1 Modified ChemXSeer	26
3.2.1.2 Ensemble extraction	27
3.2.2 ChemXSeer Tagger 2.0 - post competition	29
3.2.3 Tokenization	29

3.2.4	Extractor and features	31
3.2.4.1	Word embeddings	31
3.2.4.2	Soundex features	33
3.2.4.3	General token derived features	34
3.2.4.4	Dictionary look up	34
3.3	Results	35
3.3.1	Competition extractor - ensemble approach	35
3.3.2	Post competition - ChemXSeer Tagger 2.0	37
3.4	Conclusion	40

Chapter 4

	CiteSeerX Log Mining	41
4.1	Introduction	41
4.2	Data Processing	43
4.3	Access Analysis	46
4.3.1	Session	46
4.3.2	Downloads	49
4.3.3	Search	52
4.3.3.1	Number of document views per query	55
4.4	User Behavior Prediction	56
4.4.1	Download Prediction	56
4.5	Related Work	59
4.6	Conclusion and Future work	61

Chapter 5

	Academic Search	62
5.1	Introduction	63
5.2	Related Work	65
5.3	Academic Query Logs	67
5.4	Query Type Classification	67
5.4.1	Approach	69
5.4.2	Dataset	71
5.4.3	Experiments	71
5.5	Ranking	72
5.5.1	Dataset	75
5.5.2	Labels and Clickthrough Rates	76
5.5.3	Experiments	77
5.6	Conclusion and Future Work	81

Chapter 6	
Conclusion and Future Work	83
6.1 Future Work	84
Bibliography	86

List of Figures

2.1	The front page of Microsoft Academic Search as captured by the Internet Archive on 10 January 2013, at the time of the experiments. Note that the size of MAS is listed on the homepage.	7
2.2	An example of the list of most cited documents in MAS in Agricultural Science.	10
2.3	To estimate the number of scientific documents on the web, N , let n_0 equal the number of citations found in both Scholar and MAS for a collection of papers, and let n_g be the number of citations reported by Scholar. Then n_0/n_g is an estimate of p_m , the fraction of documents indexed by MAS. The total number of documents N would be S_m/p_m where S_m is the size of MAS.	12
2.4	Relative number of documents by scholarly search engines and databases. Total and Google Scholar are estimates. . .	13
2.5	The relative number of documents on the web for each of the 15 fields as defined by MAS.	15
3.1	Precision-Recall curves for CEM task on training dataset .	38
3.2	Precision-Recall curves for CEM task on development dataset	39
4.1	Plot of the length of the the session (number of requests in the session) against the frequency of this length in log log scale.	47
4.2	A state transition graph of web pages covering searching, viewing and downloading documents	48
4.3	Plot of the number of downloads a paper receives against the number of papers with this many downloads in log log scale.	50
4.4	The percentage of total downloads generated by papers in the top 1% and 5% respectively of most downloaded papers within a month	53

4.5	Distribution of the number of terms in a query.	54
4.6	Cache hits of the top 1% most downloaded documents for each month. Each curve represents caching policy. LR refers to logistic regression, BL refers to base line. April and May of 2012 are omitted because very low traffic was recorded in them	60
5.1	Feature importance of navigational query classification. Number of tokens does contribute the most but all other features give reasonable contributions.	75

List of Tables

2.1	The estimated number of documents on the web for each field. . . .	14
2.2	The percentage of publicly available scholarly documents found in Google Scholar.	17
3.1	Accuracy of multiple tokenizers when tested on the chemical entities of the test set	30
3.2	Similar words to calcium when sorted by cosine similarity	32
3.3	Example of word embedding clusters	32
3.4	Soundex code for English letters	33
3.5	Probability of a candidate entity conditioned on possible values of the indicator random variables for each of the three taggers used. .	36
3.6	Performance of the ensemble extractor on the CEM task at various confidence thresholds.	37
3.7	Performance of ChemXSeer 2.0 Tagger on the test dataset	38
3.8	Performance of ChemXSeer 2.0 Tagger using different collection of features	39
4.1	Percentage of traffic from the top 10 countries	50
4.2	Venues ranked by the number of downloads their papers received . .	52
4.3	Search type distribution. Blank field falls back to Document search	52
4.4	The number of result pages a user looks at when doing document search	55
4.5	Number of document views per query	55
4.6	Cash hit rate for the logistic regression model at each month, compared to the baseline and the Oracle.	59
5.1	Navigational Query Features	69
5.2	Navigational query classification performance for multiple learning algorithms. Numbers between parenthesis refer to standard deviation in 5 fold cross validation.	72

5.3	The features used in representing every query document pair . . .	74
5.4	The performance obtained at various levels by multiple feature set combinations and baselines by applying LambdaMART algorithm. Training is optimized for NDCG@10 in all cases. All - [feature] denotes using all features except for the group listed between brackets. Plus sign denotes using only the listed features.	80

Acknowledgments

First and foremost, all praise is due to Allah for blessing me with the ability to finish this dissertation. Second, I am eternally in debt to my advisor and mentor Dr. Lee Giles. His guidance, advice, and motivations were indispensable through graduate school. I am also thankful to the great senior graduate students that shared the office space with me when I first joined the CiteSeer group: PucktaData Treeratpituck, Pradeep Teregowda, and Jian Huang. The discussions and interactions with them have been a great resource in shaping my research skills. I am also grateful for my colleagues Dayu Yuan, Sujatha Das, Wenyi Huang, Zhaoui Wu, Kyle Williams, Hung-Hsuan Chen, and Jian Wu. It has been useful to discuss research problems with them. I would like also to thank the instructors who taught me at Penn State and shared their knowledge.

And to my wife, Hala, thank you very much for being by my side through this journey. Without your support and motivation this dissertation would not have been possible. Last, I would like to thank my friends in State College with whom I have shared some of the best memories of my life.

The work in this dissertation was partially funded by the National Science Foundation.

Dedication

To my parents! Without your constant support and endless belief in me, I would not have made it this far.

Chapter 1 |

Introduction

Scholarly documents are the main medium through which science is transferred and communicated between scientists, through generations, and across disciplines. It includes, but not limited to, journals, conference proceedings, technical reports, theses, and white papers. However, scholarly data is not limited to documents. A single scientific document can generate multiple collections of data that, all together, make scholarly data big. For example, tables and figures within scientific documents are important pieces of information that need to be extracted and made searchable independently. Other scientific fields introduce different kinds of data within a single document. In computer science algorithms are described as pseudo code, and are of interest to be identified on their own. For chemistry, chemists are especially interested in the formulae, molecules, and chemical names that appear within a research article. When all these sub-document objects are added together, scholarly big data is created.

As the rate at which scholarly publications are produced increases, it is inevitable that humans will fail to carry many of the tasks required to keep track of scientific production. Similarly, researchers will not be able to keep up with the latest inventions in the area of their research. Therefore it is desired to study the way by which academics access scholarly data, and explore methods to facilitate better accessibility. But before delving into enhancing scholarly data accessibility, we would like to find out more about the size of the data, and the characteristics of the users who seek it.

Recent work estimates the journal production to be around 1.8 million journal articles per year [1]. However, we do not have estimates of the total number of scholarly documents that has been published. This estimate is important for

multiple reasons. First, it is intriguing to find out how much science have we produced as humanity. And secondly, from an engineering stand point, designing search and discovery systems that facilitate access to research requires knowing the size of the collection which will be served in such a system. For if the the design phase underestimated the number of papers to be included in a search system, the quality of the service will be severely affected. Furthermore, from a scientific point of view, devising effective algorithms that enables search and discovery at scale benefits from knowing the underlying data on which the algorithms will operate. Finally, it advances the state-of-the-art for estimating sizes of collections.

Scholarly data is generated both manually and automatically. While scholarly documents are the result of human writing, much of the scholarly data, such as the sub objects, are generated automatically using extraction algorithms. Although it is possible to identify these objects manually, such efforts fail to scale typically. Automatic methods for extraction have been successfully used for citation indexing, identifying paper information such as titles authors [2], and sections within documents. These automatic methods for extraction have contributed to the birth of many digital libraries and search engines that facilitated discoverability of scholarly data, such as CiteSeer, later CiteSeerX¹, Microsoft Academic Search², and Google Scholar³.

With much of the scholarly data being generated through the automated means of information extraction, we address chemical entity extraction in chemistry scholarly documents as an example of how information extraction may be used to identify entities of interest within papers. By identifying the entities it is possible to introduce new methods for accessing them such as specific search applications, and scenarios. Furthermore, discovering chemical entities can benefit from special indexing and tokenization that is more powerful than traditional text-based search. We demonstrate how two methods can be used to extract chemical formulae and names accurately from text.

Over the past twenty years, many digital libraries, repositories and academic search engines have become popular among academics. These search engines became the starting point for conducting research as they are used to find relevant work on top of which new ideas and approaches are introduced. The interactions with

¹<http://citeseerx.ist.psu.edu>

²<http://academic.research.microsoft.com/>

³<http://scholar.google.com>

these digital libraries provide a wealth of information that captures the interests of users and the patterns they exhibit while accessing scholarly data. It is through understanding user needs and patterns that better methods for accessibility can be developed. We will be using the user access logs of CiteSeerX digital library and search engine to study their interactions with the system and infer new insights. The access logs used cover more than three years, from late 2009 to early 2013. Through this dataset we will unveil the user search and download patterns, model how papers get accessed, and how long users tend to spend on the search engine during the session.

Later, we focus on the academic search problem which is arguably the single most important aspect for scholarly documents discovery. Historically, librarian search has lead to many innovations in information retrieval such as the use of document vocabulary for indexing, and later inspired the idea behind PageRank. Because of the popularity of typical web search, many of the same ideas and technologies have found its way into academic search. However, the studies of recent academic search are rare compared with that of web search. In this work we start by studying the user query intent, which to the best of our knowledge is the first of its kind. Later we focus on learning effective ranking functions that help academics discover scholarly documents more effectively. We demonstrate how learning to rank approaches can be applied to achieve good performance.

In the next few sections we give a brief introduction to each of the sub problems that are tackled in this dissertation.

1.1 Estimate of the Number of Scholarly Documents on the Web

The first problem introduced and tackled in this dissertation is obtaining an estimate on the total number of scholarly documents available on the web. For convenience, we will refer to all academic and scientific documents as “scholarly”. By scholarly documents, we mean journal and conference papers, dissertations and masters theses, books, technical reports and working papers. Patents are excluded.

The web has become a standard resource for such documents because individual authors, academic and research publishers, and repositories have made their

documents available online, with some open to the public and others limited to subscribers.

Capture-recapture methods are used to obtain an estimate using two major academic search engines, Google Scholar and Microsoft Academic. Each search engine is assumed to be a snapshot of the population at a certain time, and using the intersection of two search engines an estimate is derived. Furthermore, size estimates for individual scientific fields are computed by reinterpreting the experiments at the micro level.

The focus is later shifted towards estimating the percentage of publicly available scholarly documents on the web. These documents are available at no charge without the need of subscription or payment of any kind. Such estimate is crucial to digital libraries and repositories such as CiteSeerX which crawl the web looking for scholarly articles. It is far more important for policy makers looking and working on implementing open access policies to scientific literature that was funded by government organizations.

1.2 Chemical Entity Extraction

Identifying chemical formulae and chemical names inside chemistry research articles is the first step for many applications including search and discovery. Besides, the mentions of chemical entities, hereafter chemical entity refer to chemical formulae and chemical names, comprise a large collection of data that construct part of the scholarly big data.

In this section we introduce two methods for identifying chemical entities in chemistry literature. One approach utilizes a group of information extraction systems, while the other is a standalone approach. The first approach is suitable when multiple extractors are provided, and their results need to be merged to outperform the individual system's in terms of accuracy. The latter method introduces novel feature sets that identify chemical entities using a Conditional Random Fields (CRF) extractor.

1.3 Log Analysis of CiteSeerX

Access logs of CiteSeerX search engine and digital library from the period of September 2009 to March 2013 are analyzed. First bots access logs are filtered out to keep only requests from real users using multiple heuristics. Later sessions are inferred, and the number of clicks per session is modeled where it is found that a power law distribution is a good fit. The number of downloads a given paper receives is studied where it is also found that a power law distribution would be a good approximation.

In our analysis we observe that highly accessed papers in a given month are likely to remain highly accessed in the following months. Therefore, we use a logistic regression model to predict the highly accessed documents for a given month using historical data. This prediction allows the system to cache these highly accessed documents, thus providing faster turn around time to the users in addition to reducing resource usage.

1.4 Academic Documents Search

In this problem we first start by classifying the academic search queries based on the user intent into two major classes: navigational, and informational. Both classes of queries are defined. For navigational queries, the multiple facets that makes a query navigational are listed. Boosted decision trees are then used with a carefully crafted set of features to identify navigational queries from informational ones.

After that we focus on devising better ranking functions for academic search. We first create a dataset of queries along with relevance judgement for documents that match each of the queries. The dataset is used to examine the relation between absolute click through counts and documents relevance. After that we use a learning to rank approach that utilizes multiple features and relevance models to learn a ranking function using the manually labeled relevance judgments. At the end, the contribution of some of the feature sets is examined where it is found that article full text is one of the most informative features.

Chapter 2 |

The Number of Scholarly Documents on the Public Web

2.1 Introduction

Many researchers and academics are concerned about the extent to which academic and scientific documents are available on the web, as well as their ability to access them. For convenience, we will refer to all academic and scientific documents as “scholarly”. By scholarly documents, we mean journal and conference papers, dissertations and masters theses, books, technical reports and working papers. Patents are excluded.

The web has become a standard resource for such documents because individual authors, academic and research publishers, and repositories have made their documents available online, with some open to the public and others limited to subscribers.

Numerous databases and search engines such as Google Scholar and CiteSeer track scholarly documents and thus facilitate research. However, the coverage of some of these search engines and databases is unknown. An important question that a scholar or researcher might ask is whether a single search engine or database is sufficient to obtain comprehensive results in a particular field. For example, *Web of Science* reported that as of January 2013 it comprises more than 49.4 million records [3], and Microsoft Academic Search (MAS) stated that it covers 48.7 million documents [4] (A snapshot of the main page of MAS which used to show this statistic is shown in Figure 2.1). However the size of Google Scholar is unknown

despite studies that have tried to determine the extent to which Scholar's citations overlap with those of other citation indices [5,6]. Relatively smaller digital libraries and databases, such as CiteSeer and PubMed, tend to focus on documents from certain fields, most of which are also indexed by large search engines such as Google Scholar and MAS. Bjork et al. [7] estimated the number of published papers in 2006 to be roughly 1.35 million, whereas a similar estimate for 2011 put the number at 1.8 million [1]. But despite the availability of per year estimates, researchers have yet to provide an estimate of the total number of published scholarly documents.

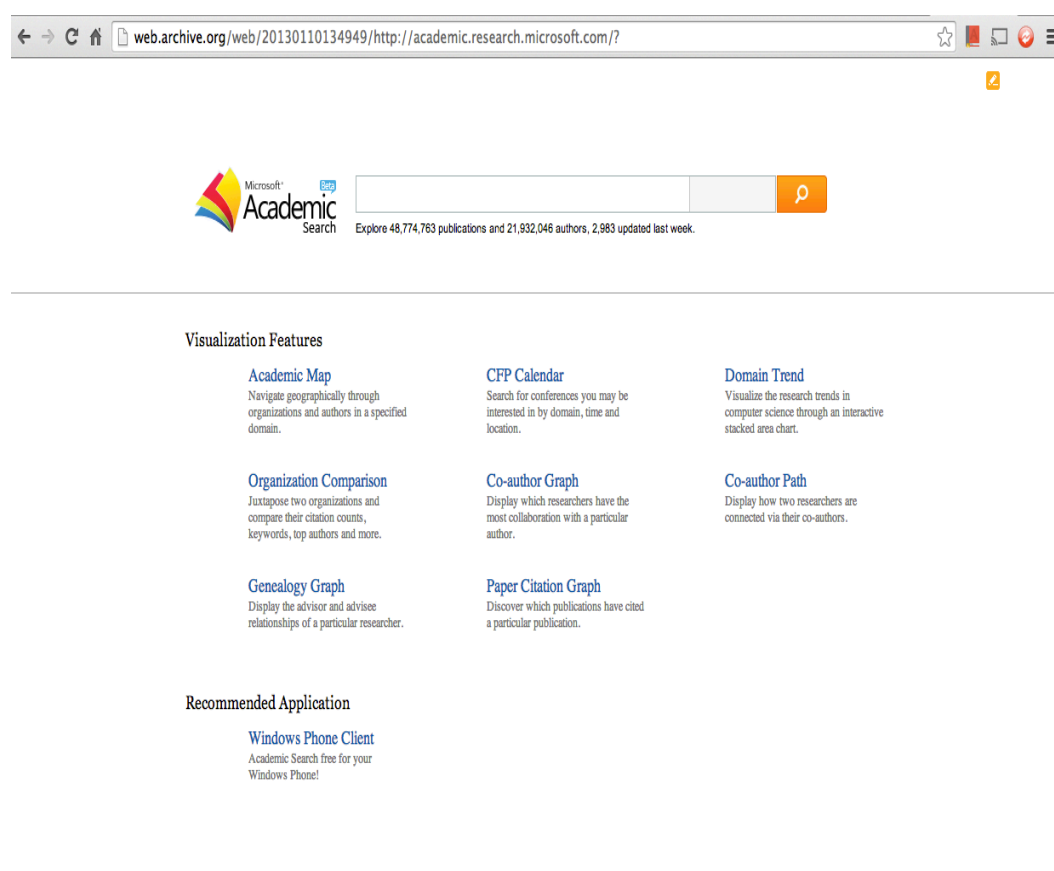


Figure 2.1. The front page of Microsoft Academic Search as captured by the Internet Archive on 10 January 2013, at the time of the experiments. Note that the size of MAS is listed on the homepage.

Estimating the number of scholarly documents available on the web is quite different from estimating the size of the web itself, and thus presents different challenges. Studies that offer estimates of the size of the web such as Lawrence and

Giles [8,9], Bharat and Broader [10], or Dobra and Fienberg [11] can not be used to estimate the number of scholarly documents on the web for many reasons. For example, search engines are no longer receptive to automated requests for fear of denial of service attacks or reverse engineering of their ranking function. Checking that a document indexed by search engine A is also available in the index of search engine B is nontrivial. To estimate the size of the web, one strategy would be to check whether a particular URL is available in both engines. However, in the case of scholarly documents the search engines might not have obtained their copies from the same location since the same document might be available at different URLs. Therefore, it is necessary to explore the content of the document and not just the location from which it was obtained. Even when a search engine returns the location of a certain document, it could be that the publisher offers full access to subscribers only and has a limit on the number of downloads allowed per day, thus making automated methods impractical. Finally, many publishers restrict access for many web crawlers.

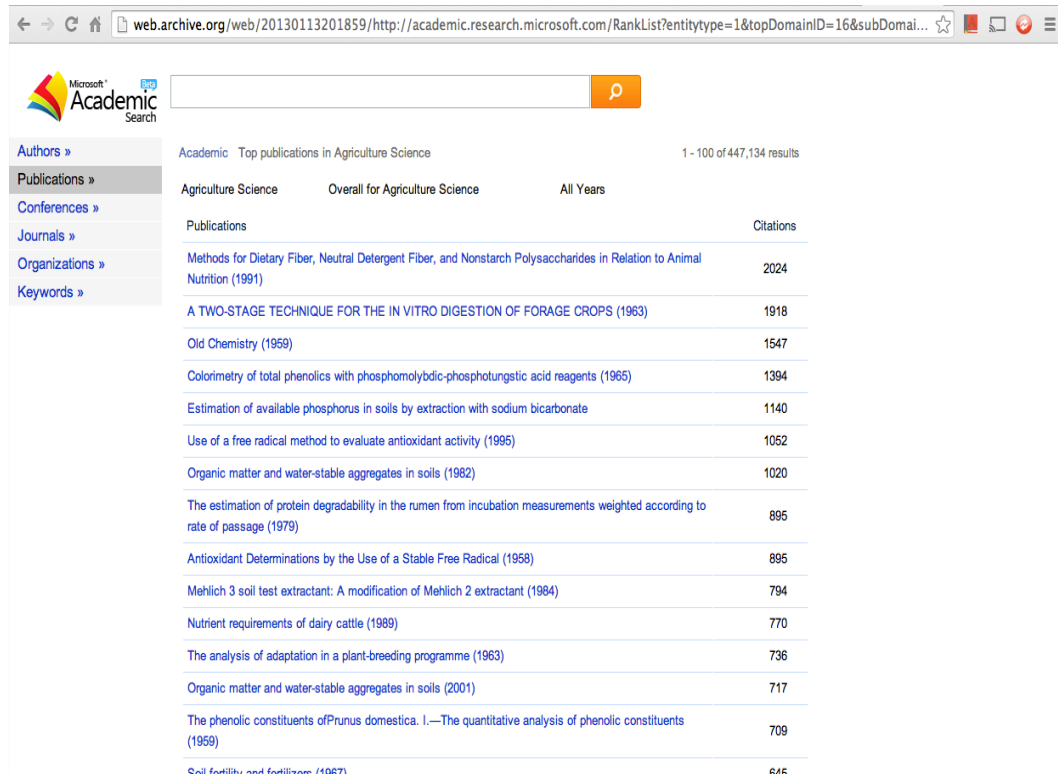
2.2 Estimating the Number of Scholarly Documents on the Web

To estimate the number of scholarly documents on the web, we use the relative size of two major academic search engines: Google Scholar (Scholar) and Microsoft Academic Search (MAS). We note that our estimates are limited to English documents only. We used the option offered by Google Scholar of filtering results by language, whereas for MAS we ran a language detection algorithm on the title of each document. Only those identified as English were used. Our approach can be described as follows. Assuming that each academic search engine would sample the web independently for papers, then each index would contain a subset of available documents. Next, we considered each search engine to be a random capture of the document population at a certain time. Using the intersection of these two captures, we estimate the entire size of the population. However, since obtaining the database of both academic search engines was not feasible, we approximated the overlap by randomly sampling from each search engine and then determining the size of overlap in the random sample. The simplest approach for sampling from two search

engines is to send queries to each and then measure the overlap of the results. This approach was used by Lawrence and Giles [8,9] and by Bharat and Broader [10]. However, it is known to suffer from many biases and statistical dependencies. To mitigate the effect of bias and dependence and to obtain a selection that was as random as possible, we sampled from each academic search engine with the following methodology: if we choose a random paper p that is in the database of an academic search engine, then the set of papers S that cite p is a random collection from this search engine. If we collect the set of papers citing p from both Google Scholar and MAS, then the overlap between these two is an estimate of the overlap between the two search engines. This method provides a good estimate of the coverage of each search engine because when an academic search engine builds its database by indexing a new document, it has no knowledge of the incoming citations to this document. Therefore, the search engine has to obtain all the available manuscripts and analyze them in order to determine whether there are any citations to a target paper. In contrast to references, which the search engine can extract from the document and try to obtain a copy of each referenced item, incoming citations are not embedded with a document. Hence, to build a complete citation network, it is necessary for a search engine to obtain all the available scholarly documents. The more documents the search engine obtains, the larger its citation network.

Based on the methodology described, we chose 10 documents from each of the fifteen fields specified by Microsoft Academic Search: Agriculture Science, Arts and Humanities, Biology, Chemistry, Computer Science, Economics and Business, Engineering, Environmental Sciences, Geosciences, Material Science, Mathematics, Medicine, Physics, Social Sciences, and Multidisciplinary. The list of papers used as queries for which we retrieved the collection of incoming citations was randomly chosen from the most cited documents in each field. Special care was taken in regard to choosing documents because search engines impose a limit on the maximum number of retrievable results. Therefore, the chosen documents each had fewer than 1,000 citations in Scholar and likewise fewer than 1,000 citations in MAS. Figure 2.2 shows an example of a page that contains the most cited papers in agricultural science, according to MAS as of January 2013.

The experiments were performed during the period of January 10-12, 2013 by, (1) sending 150 requests to each search engine requesting the list of incoming citations to each paper such that each request corresponds to one paper, and (2) storing the



Microsoft Academic Search

Academic Top publications in Agriculture Science 1 - 100 of 447,134 results

Publications	Citations
Methods for Dietary Fiber, Neutral Detergent Fiber, and Nonstarch Polysaccharides in Relation to Animal Nutrition (1991)	2024
A TWO-STAGE TECHNIQUE FOR THE IN VITRO DIGESTION OF FORAGE CROPS (1963)	1918
Old Chemistry (1969)	1547
Colorimetry of total phenolics with phosphomolybdic-phosphotungstic acid reagents (1965)	1394
Estimation of available phosphorus in soils by extraction with sodium bicarbonate	1140
Use of a free radical method to evaluate antioxidant activity (1995)	1052
Organic matter and water-stable aggregates in soils (1982)	1020
The estimation of protein degradability in the rumen from incubation measurements weighted according to rate of passage (1979)	895
Antioxidant Determinations by the Use of a Stable Free Radical (1958)	895
Mehlich 3 soil test extractant: A modification of Mehlich 2 extractant (1984)	794
Nutrient requirements of dairy cattle (1989)	770
The analysis of adaptation in a plant-breeding programme (1963)	736
Organic matter and water-stable aggregates in soils (2001)	717
The phenolic constituents of <i>Prunus domestica</i> . I.—The quantitative analysis of phenolic constituents (1959)	709
Soil fertility and fertilization (1967)	645

Figure 2.2. An example of the list of most cited documents in MAS in Agricultural Science.

returned metadata about each citation which included the document’s title, list of authors, number of citations, year of publications, and the venue of publication (if available). Overall, we obtained 41,778 citations from MAS and 86,870 citations from Google Scholar. Matching the citations across results from different sources (Scholar and MAS) cannot be achieved solely on the basis of verbatim matching of title and authors. The reason is that academic search engines obtain their metadata in different ways. For example, a publisher might provide some or all of the metadata. Alternatively, the metadata of the document might be automatically extracted from a downloaded document from the web. In the latter case, errors are inevitably introduced in the extraction stage resulting in noisy metadata. Though we have no way of establishing whether a certain paper’s metadata was provided by a publisher or automatically extracted, we found evidence that the results are mix of both cases. Another issue was that Scholar and MAS differed occasionally in terms of their respective result encoding, especially with regard to Latin letters.

Therefore, the records returned by MAS and Scholar for a given paper were matched as follows. To match the Scholar citation collection C_g with the MAS citation collection C_m , for the same paper, we first matched each paper in C_g with its counterpart in C_m such that the papers' titles were exact textual match. Later, we constructed shingles of size two for all the titles in both C_g and C_m . The collection of size two shingles for a title is the set containing every two continuous words appearing in that title [12]. For example, the size two shingles for the sentence: "A Brief History of Time" would be $\{A\ Brief, Brief\ History, History\ of, of\ Time\}$. Given the set of shingles S_1 for a paper in C_m , and the set S_2 for a paper appearing in C_g , we computed Jaccard similarity between S_1 and S_2 as follows:

$$Similarity(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

We computed the Jaccard similarity between every pair of documents appearing in C_g and C_m , and considered a pair S_1, S_2 to be a match when their similarity was above a certain threshold. Based on our experiments with different values of the threshold for accuracy, we empirically selected 0.50. After matching the collections C_g and C_m as described, we manually evaluated the matched records individually for mistakes. We found mistakes in less than 2% of the matched records, and all false negatives and positives were corrected. Overall, more than 4,000 record pairs were manually inspected.

We computed the overlap between the results for all the 150 query documents, and measured the total number of unique documents that cited the query documents. The overall size of scholarly documents on the web can be estimated using capture/recapture (refer to the supplementary material for an introduction to capture/recapture). Assuming that the total number of documents on the web is N , and each search engine samples the web independently, then the quantity n_0/n_s where n_0 is the number of documents returned by both Scholar and MAS, and n_s is the number of documents returned by Scholar is an estimate of the fraction of scholarly documents, p_m , indexed by MAS. Then, the number of documents on the web N can be estimated as s_m/p_m where s_m is the number of documents indexed by MAS. These variables are illustrated in Figure 1. At the time of this study, s_m was listed as 48,774,763 by MAS. However, according to our analysis 98% of the returned papers from MAS were found to be in English. Therefore, in

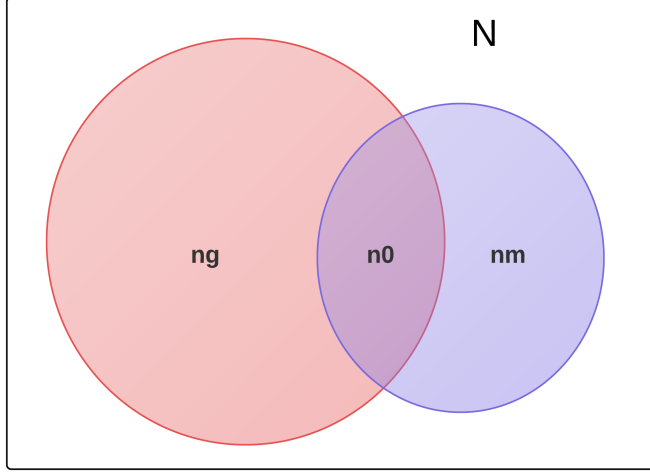


Figure 2.3. To estimate the number of scientific documents on the web, N , let n_0 equal the number of citations found in both Scholar and MAS for a collection of papers, and let n_g be the number of citations reported by Scholar. Then n_0/n_g is an estimate of p_m , the fraction of documents indexed by MAS. The total number of documents N would be S_m/p_m where S_m is the size of MAS.

our estimates we used $0.98 * 48774764 = 47799267$ as an estimate of the number of English papers in MAS, s_m . Next, p_m was estimated to be 0.418, yielding an estimate size of N , the total number of documents on the web, of 114,000,000.

We argue that this estimate is a lower bound of the number of scholarly documents on the web because the likelihood that a document is in an academic search engine given that it was found in another academic search engine, is larger than the likelihood that any given document is indexed by an academic search engine. Although we designed our experiments to mitigate any possible statistical dependence by relying on citations instead of query results, the experiments do introduce a bias against documents with more than 1,000 citations. Search engines impose a restriction on the number of retrievable results for all type of queries, unless an Application Programmable Interface (API) is provided. Hence, any study based on sampling from a search engine, regardless of the approach, would encounter this bias. For our study it is relevant to note that Google Scholar at this time does not provide an API.

Using the statistics calculated above, we estimated Google Scholar to have 99.3 million documents, which is, approximately, 87% of the total number of scholarly

documents found on the web. This percentage is close to the 86% reported by Norris, Oppenheim and Rowland [13] when they tested the coverage of Google and Google Scholar for finding Open Access documents. With this estimated size, Google Scholar is more than twice as large as the nearest alternative, as MAS and Web of Science are both reported to have fewer than 50 million records. However, we estimate that Scholar fails to index 13% of all web accessible documents. This implies that it is necessary to search across multiple search engines in order to retrieve a comprehensive list of results. The relative size of each database/search engine is depicted in Figure 2.4.

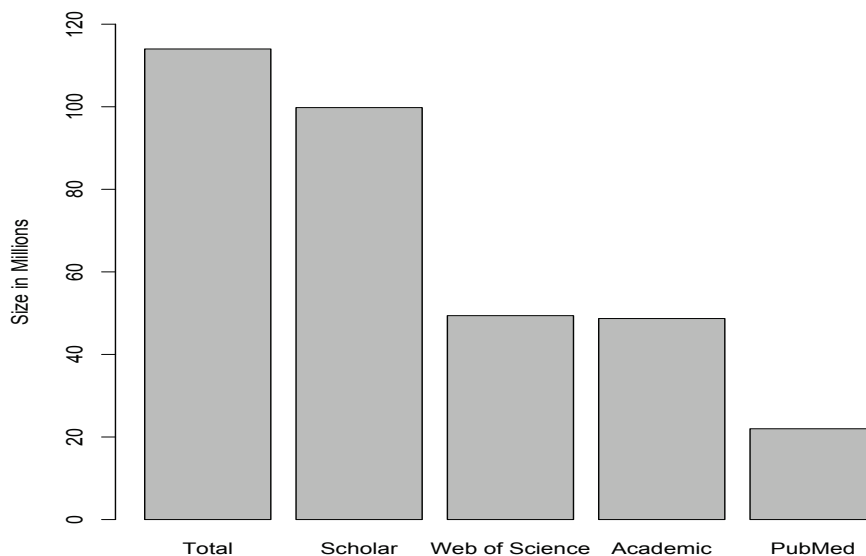


Figure 2.4. Relative number of documents by scholarly search engines and databases. Total and Google Scholar are estimates.

2.3 Field Level Analysis

In addition to computing statistics about the total number of scholarly documents on the web, we can reinterpret the experiments at the field-scale, making it possible to obtain estimates of the size of each of the fifteen scholarly fields defined in MAS. To obtain these estimates, we assumed that a paper and its citations belonged to

Table 2.1. The estimated number of documents on the web for each field.

Discipline	Size in MAS	Estimate of Size #1	Estimate of Size #2
Agriculture Science	447,134	1,088,711	1,026,904
Arts & Humanities	1,373,959	5,286,355	3,155,485
Biology	4,135,959	8,019,640	9,498,798
Chemistry	4,428,253	10,704,454	10,170,091
Computer Science	3,555,837	6,912,148	8,166,468
Economics & Business	1,019,038	2,733,855	2,340,360
Engineering	3,683,363	7,947,425	8,459,349
Environmental Sciences	461,653	975,211	1,060,249
Geosciences	1,306,307	2,302,957	3,000,113
Material Science	913,853	3,062,641	2,098,789
Mathematics	1,207,412	2,634,321	2,772,987
Medicine	12,056,840	24,652,433	27,690,190
Physics	5,012,733	13,033,269	11,512,430
Social Science	1,928,477	6,072,285	4,429,012
Multidisciplinary	9,648,534	25,798,026	22,159,184
Total Sum		121,223,731	117,540,415

the same field. Though this assumption does not always hold, we assumed that it would be a good approximation to the number of citations within a discipline. We also noted that it is possible for some papers to be classified into multiple fields especially in closely related fields, e.g. engineering and mathematics. Nevertheless, as the number of citations grew for a given paper, we anticipated more papers from the same field would cite it.

Using the classification provided by MAS, and the number of papers reported in each field, we used the 10 queries in the experiments for each field to compute the overlap between Scholar and MAS in that particular field. Table 2.1 reports the estimate of the total number of available documents using the procedure described above (method #1 in the table).

The sum of the individual field estimates yields a total of 121 million, (last row of Table 2.1) which is close to the 114 million estimate obtained earlier for the total number of documents across all fields. This supports our assumption that the per field estimate is fairly accurate, and is not much affected by cross field references or the chance of assigning a paper to multiple fields. Hence, the numbers estimated in Table 1 are indicative of the actual size of each field. The relative size

of each field is shown in the pie chart in Figure 2.5. In addition to computing a capture/recapture estimate for the size of each field, we report on another method for this estimate. In this approach, each field's size is computed as the percentage of total available documents on the web based on our previous estimate of 114 million for the total number of scholarly documents. The percentage is obtained by computing the field's percentage size in MAS. For example, MAS is reported to have 4,135,959 documents in biology. Therefore the percentage of biology to the total number of scholarly documents is $\frac{Size(Biology)}{Size(MAS)} = \frac{4,135,959}{48,774,763} = 0.08$. Thus, with this method (method #2), the estimate of the total number of documents in biology is $0.08 * 114,000,000 = 9.6$ million. We notice here that the assumption of citations belonging to the same field is under sampling certain fields, while over sampling others. However, it is quite close to the percentage-based estimate in many fields.

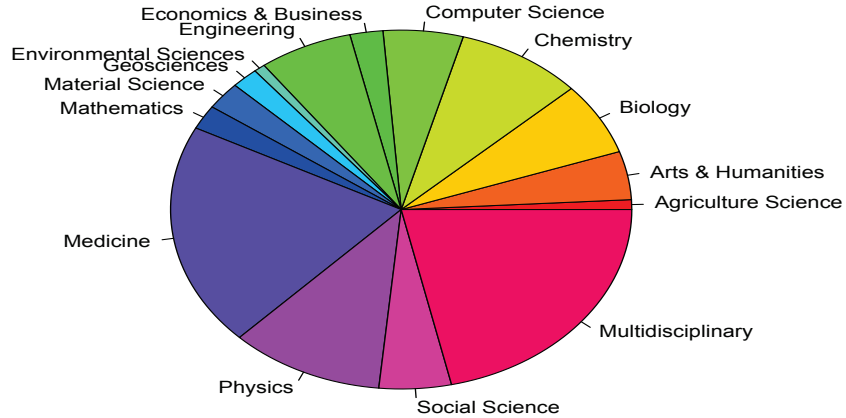


Figure 2.5. The relative number of documents on the web for each of the 15 fields as defined by MAS.

Another interesting estimate is the percentage of scholarly documents on the web that are freely available, i.e. can be accessed without paying a fee or needing a subscription. We used Google Scholar to estimate this percentage because Scholar

provides a direct link to the publicly available document next to each search result where a link is available. Note that there is no easy way to distinguish between publisher’s links and public links in MAS. As our estimate found that Scholar contains only 87% of the available scholarly documents on the web, our estimate of the percentage of public documents is limited to the coverage of Scholar. However, this is still a good indicator of the relative availability of publicly available documents. To estimate the percentage of publicly available documents for each field, we randomly sampled 100 documents from MAS belonging to each field such that each document had at least one citation. We imposed a citation limit to filter out documents that are collected by MAS that were not real scholarly documents (although it is rare to find such documents, they nevertheless exist). Then, each of the 100 documents was searched on Google Scholar to establish whether the document was freely available on any site. The percentage of freely available documents for each field is reported in Table 2.2. In the last two columns, we multiply the estimate of the percentage of freely available documents by the size estimate of the field in Table 2.1 (method #1), resulting in the total number of freely available documents in that field.

The 95% one sided lower bound confidence interval for the estimated number of freely available documents is 27.8 million, which accounts for roughly 24% of the total estimate of scholarly documents. This estimate is a weighted average of the one sided 95% lower bound confidence interval of the percentage of freely available documents in each field multiplied by the respective estimated field size. The lower end of the one sided confidence interval is for the proportion size, and is computed as:

$$\frac{y}{n} - z_{\alpha} \sqrt{\frac{(y/n)(1 - y/n)}{n}}$$

where y is the number of publicly found documents, $n = 100$, and $z_{\alpha} = 1.645$ is the standard normal distribution at $\alpha = 0.95$ [14].

It would be interesting, however, to determine the quality of these freely available documents. It is also worth pointing out that this estimate of 24% for the percentage of publicly accessible scholarly documents is a bit higher than the 15-20% documents estimated to be self-archived [15, 16].

Note here that our sampling is uniform, because we retrieved the document

Table 2.2. The percentage of publicly available scholarly documents found in Google Scholar.

Field	% of Public	95% CI	Estimate of Size	95% Lower Bound
Agriculture Science	12	± 6.3	130,645	72,446
Arts & Humanities	24	± 8.3	1,268,725	897,331
Biology	25	± 8.4	2,004,910	1,433,666
Chemistry	22	± 8.1	2,354,979	1,625,540
Computer Science	50	± 9.8	3,456,074	2,887,549
Economics & Business	42	± 9.6	1,148,219	926,256
Engineering	12	± 6.3	953,691	528,852
Environmental Sciences	29	± 8.8	282,811	210,017
Geosciences	35	± 9.3	806,034	625,341
Material Science	12	± 6.3	367,516	203,799
Mathematics	27	± 8.7	711,266	518,878
Medicine	26	± 8.5	6,409,632	4,630,828
Physics	35	± 9.3	4,561,644	3,539,034
Social Science	19	± 7.6	1,153,734	761,868
Multidisciplinary	43	± 9.7	11,093,151	8,992,160
Total			36,703,036	27,853,573

IDs of all the documents in each given field from MAS, then uniformly chose 100 that conformed to the citation sampling restriction. To the best of our knowledge, this is the only uniform sampling method for estimating the percentage of freely available scholarly documents. The numbers reported in Table 2 differ from other recent estimates in regard to the number of documents available on the web as open access, e.g. Bjork et. al. [17]. We believe this difference arises from the sources from which they sampled. For the other recent estimate researchers considered only journals over the period of one year, whereas our definition of scholarly documents is not limited to journals and sampling was cumulative, i.e. not limited to any time period. Compared to other kinds of publications, journal publications are more likely to be indexed by databases such as Web of Science [7]. However, other documents such as conference proceedings and technical reports, though influential may not be indexed by Web of Science. As an example, the famous PageRank paper [18], which presents the seminal algorithm for Google ranking was published as a technical report. Therefore, Web of Science does not index it.

2.4 Conclusion

In summary, the lower bound estimate of the number of scholarly documents, published in English, available on the web is roughly 114 million, of which Google Scholar covers nearly 87%, approximately 100 million documents. Therefore, it would be useful for researchers to consider as a standard practice querying multiple databases and academic search engines in order to gain the most comprehensive result for their query. Also, we estimate that almost 1 in 4 of web accessible scholarly documents are freely and publicly available. Our estimates for specific academic fields differs significantly, such that some fields have 4 times greater percentage of freely available documents than others.

2.5 Additional Estimates

This section reports how we estimated the number of scholarly documents on the web using another method based on Poisson regression capture/recapture. The estimates obtained using this method are similar to the estimates reported earlier and provide validation for our approach.

The approach we used to estimate the total number of scholarly documents on the web is usually reported in the literature as the Lincoln-Petersen method [19,20]. It also happens to be the maximum likelihood estimator of a population size when the captures are modeled as a hypergeometric distribution [21]. These methods are based on the assumption that the population size does not change between captures, and that the probability of capturing an object in the later captures does not change after the first capture. Assume that n_1 items were captured, then released after being labeled. A second capture results in n_2 items, of which m are labeled from the first capture. Then, the estimate of the population size N is given by

$$\hat{N} = \frac{n_1 n_2}{m}$$

In our experiment for estimating the total number of scholarly documents, the assumption of a closed population size is not completely preserved because search engines constantly add new documents. But since we confined the time window of

the experiment to two days, and given that the number of citations grows slowly (unlike news articles), we argue that the assumption of a closed population is a reasonable approximation. However, as we previously point out, the conditional probability of search engine B capturing a document previously captured by a search engine A is larger than or equal to the probability of capturing a document by B . This is primarily due to the tendency of web crawlers to index pages that are connected to other pages, i.e. more popular pages (a similar argument was made by Lawrence and Giles [8]). Although in practice the capture probabilities differ between captures, we argue that this approach produces a good estimate of the total population size, and a good approximation of the total number of scholarly documents on the web.

To test the validity of this assumption, we used an estimate that allows the probability of capturing items across different occasions to vary. As such, the probability of capturing an item i on the second capture can differ from the probability of capturing i at the first capture. Next, with *Rcapture* [22], Poisson regression was used to estimate the parameters of the capture/recapture model [23–25]. Using Poisson regression, the estimate of the population size as computed by *Rcapture*, while allowing for variability with regard to capture probability across time, would put the percentage of scholarly documents covered by MAS at 0.418, hence estimating the number of scholarly documents on the web to be 114 million. This is exactly the Maximum Likelihood Estimator value obtained using the Lincoln-Petersen method. And it is also the approach with the lowest Akaike Information Criterion (AIC) [26], a measure of relative goodness of fit. Note that the lower the value of the AIC, in this case 41, the better fit of the data. A 95% confidence interval of the population size results in a coverage percentage for MAS in range of (0.416,0.419). However, if the probability by which items are captured is assumed to be the same across the captures, then the Poisson regression model would estimate a population size that would put the coverage percentage of MAS at 0.366. Therefore, the total number of scholarly documents on the web would be estimated as 130 million. This approach, on the other hand, has a high AIC of 41914 which indicates poor goodness of fit compared with the varying capture probabilities model.

In conclusion, our experiments are supported by two methods for estimating the document population. As both methods obtain the same estimate, we reported earlier the simpler method based on the maximum likelihood estimator, Lincoln-

Petersen.

Chapter 3 |

Chemical Entity Extraction

In this chapter we describe an example of sub-objects that are extracted from scholarly documents that make scholarly data big. We focus on chemistry documents, and describe methods for extracting chemical formulas and chemical names.

We have been witnessing a great interest in identifying and extracting chemical entities in academic articles, with many approaches being proposed. Each of the proposed methods, spanning from rule based systems to machine learning approaches, has its own strengths and shortcomings. Here, we describe a probabilistic framework that allows for the output of multiple information extraction systems to be combined in a systematic way. The identified entities are assigned a probability score that reflects the extractors' confidence, without the need for each individual extractor to generate a probability score. Previously, as part of the CHEMDNER challenge, we used an ensemble of extractors by combining the output of three prominent extractors. Later, we quantitatively compared the performance of multiple chemical tokenizers to measure the effect of tokenization on extraction accuracy. Then, a single Conditional Random Fields (CRF) extractor that utilizes the best performing tokenizer is built using a unique collection of features such as word embeddings and Soundex codes.

The ensemble of multiple extractors outperforms each extractor's individual performance during the CHEMDNER challenge. When the performance was optimized to favor recall, the ensemble approach achieved the second highest recall on unseen entities. As for the single CRF model with novel features, the extractor achieves an F1 score of 83.3% on the test set, without any post processing or abbreviation matching.

3.1 Introduction and Related Work

Automatically extracting information from free text has been of interest in many fields because the task is too labor intensive to be carried by humans at scale. Some of the applications try to identify person names and locations that appear in news articles, extract product information from online retailers website, and identify author and title information in scientific publication. Scientists have their share of interest in information extraction. In the case of chemists, they are interested in identifying chemical entities appearing in scientific publications and internal reports of companies. Identifying mentions of chemical entities is crucial for the follow up task of indexing. Some of the leading companies in the chemical domain employ teams that manually identify all chemical entities appearing in their internal reports. Should identifying chemical entities become feasible automatically with high accuracy, the extraction tasks can be done on faster pace, and cover larger collections of documents.

Many approaches have been proposed to tackle the problem of automatically extracting information from free text. The simplest methods relied on dictionaries, sometimes referred to as *gazetteers*, that are compiled by domain experts. Computer programs would check if the document contains terms that are found in the dictionary; for these terms would be extracted as entities. It is, however, challenging to keep a comprehensive dictionary that contains all the entities of interest. Furthermore, dictionary based approaches can not identify new terms that are not in the dictionary. In other words, they fail to generalize beyond the list of chosen terms.

Rule based systems were later introduced as a step forward from dictionaries. Rule based information extraction systems define a set of rules that *identify* mentions of the entity type of interest. The rules are carefully crafted to capture descriptors of the entity that can be used to describe terms of a given entity type. For example, a rule for identifying chemical formulas might be that the term has no two consecutive small case letters. Rules can be cascaded and joined with dictionary look up where the term is inspected for existence in a given dictionary. Although rules provide a way to generalize beyond dictionaries and identify terms that have not been seen before, it is daunting to keep the rules up to date. In fact, crafting the rules in the first place is a very challenging task in many domains. Efforts were made to craft

the rules automatically using Inductive Logic Programming (ILP) [27, 28] .

Machine learning methods for information extraction became prevalent in the late 1990s and early 2000s with the introduction of Maximum Entropy Markov Models (MEMM) [29] and Conditional Random Fields (CRF) [30]. Both algorithms were applied successfully on information extraction tasks, achieving improvements in precision and recall over existing approaches. It seems as if CRF has taken over other methods for *sequence labeling* (information extraction is an application of sequence labeling, where the sequence is a sentence and the labels are the classes of interest that a word can take on) mainly because it solves the *label bias* problem [30] that other approaches suffer from, including MEMM. However, in many scenarios CRF models do not outperform MEMM models because the large number of parameters that need to be learned in CRFs might lead to overfitting [31].

In the chemical domain, applications of information extraction include identifying chemical formulas and chemical names appearing in research articles and technical reports. OSCAR3 [32] was one of the early systems that used automatic information extraction to extract chemical names and formulas from free text. After that, many researchers introduced new methods and tools for extracting and indexing chemical entities [33–37].

One of the main challenges in chemical information extraction was the lack of annotated corpus in which mentions of chemical entities within documents are identified by human experts. Building an annotated corpus is expensive, and can be challenging to share as the underlying documents might not be copyright-friendly. Therefore, many research teams resort to creating their own annotated corpus on which machine learning extraction algorithms were trained and tested. This, however, has made fairly comparing extraction approaches and systems challenging because it is hard to isolate the source of improvement as it might have come from the approach itself, the features used in the learning algorithm, or the quality of the training corpus.

This obstacle was mainly addressed in the BioCreative’s CHEMDNER task with the release of an annotated corpus of 10,000 PubMed abstracts. The CHEMDNER challenge had two tasks. The first is to identify all chemical formulas and names mentioned within abstracts of selected PubMed records. Formally it is referred to as the Chemical Entity Mention (CEM) task. The second is Chemical Document Indexing (CDI) which requires participants to rank all the unique chemical names

and formulas within each document. The 10,000 PubMed abstracts were divided into 3,500 abstracts for training, 3,500 abstracts for development, and 3,000 for testing. For details about the CHEMDNER task, please refer to [38].

We have participated in both tasks of the CHEMDNER challenge by submitting 5 runs for each task [39]. For the challenge, our approach involves an ensemble approach that utilized multiple *off-the-shelf* extractors and allowed for combining their output in a probabilistic fashion. The ensemble framework assigns a probability score to each extracted entity that depends on which extractors have identified or failed to identify a given term as an entity. The individual extractors that were used in the ensemble approach were: OSCAR4 [36,40], ChemSpot [37,41], and a modified version of ChemXSeer formula and name tagger [33,34]. The probability score was used later as a confidence measure that allowed for optimizing the extraction result in respect to either precision or recall. When a balanced cut off point was selected for confidence, the F1 score is optimized. This paper discusses our contribution in the CEM task only. For our experiments and results on CDI please refer to [39].

After the end of the competition we revisited the challenge tasks to investigate potential sources of improvement. We started by studying the effect of tokenization on the accuracy of the extractor. The performance of three prominent tokenizers was studied where it was found that OSCAR4 had the highest accuracy in tokenizing the test set of PubMed abstracts. Later, a new Conditional Random Fields extractor was designed using a unique collection of features that utilizes state of the art word embedding algorithms along with *Soundex* code of each term. Soundex is an algorithm that is usually used by the USA Census Bureau to encode surnames *phonetically*. The generated code contains the first letter of the surname combined with three digits representing the the last name *phonetically* (how do they *sound*). It is used for matching names with multiple spellings. The rationale here for borrowing this last name matching algorithm is that many chemical names tend to *sound* similar, albeit being spelled and structured differently. To the best of our knowledge, this is the first work that utilizes *Soundex* code to identify chemical entities.

Using the new extractor and the features, we are able to achieve F1 score of 83.3% on the test dataset without doing any post processing on the tags. In the challenge task, most of the top performing teams carried a post processing step that was optimized for this dataset only. As we are interested in building

a *general* extractor, we did not do any post processing to boost the F1 score. The CHEMDNER annotation guideline lists abbreviations to chemical entities as valid entity. However, we did not implement or tackle the problem of identifying abbreviations because we believe it is a different problem that can be solved with existing third party solutions such as [42]. Given that the best performing team obtained an F1 score of 88% using multiple extractors and sophisticated post processing and abbreviation matching algorithms, we believe our system achieves a very competitive F1 score as a standalone system that can be easily used as an API or as a program. The source code has been released on Github: <https://github.com/SeerLabs/chemxseer-tagger>

3.2 Methods

The Methods is split into two sections. In the first section, the approach used during the competition is described in detail, while the second section describes the new extractor created after the competition which we call ChemXSeer Tagger 2.0.

3.2.1 Ensemble extraction - pre competition

Our interest was initially in the *Chemical Entity Mention* (CEM) task as it is the prerequisite to the following *Chemical Document Indexing* (CDI) task. We started by running a distribution of ChemXSeer’s formula and name extractor [33–35] that is released for the general public on the training and development datasets. The tagger is based on Conditional Random Fields (CRF) [30] models, with additional rules for pre and post processing documents. The *off-the-shelf* ChemXSeer tagger was originally trained on a subset of papers from the *Royal Society of Chemistry* (RSC). The extractor performed well when evaluated on precision, but the mediocre recall ended up penalizing the overall F1 score.

We then explored the use of other open source and free chemical information extraction systems, in particular OSCAR4 [36,40], ChemSpot [37,41], Reflect [43], Whatizit [44], and MiniChem [45], on the *BioCreative* dataset. ChemSpot [37] and OSCAR4 (Originally we used the standard setting of OSCAR4, but experiments with the PubMed setting yielded similar results) [36] were promising since the former’s result had high F1 score, and the latter’s reported high recall compared

to the other extractors. To balance the performance and boost both the precision and the recall, we chose two paths to explore. First, modify ChemXSeer’s tagger and retrain it on the corpus at hand since the distribution of vocabulary might be different in the *BioCreative* dataset from the original Royal Society of Chemistry (RSC) articles that were used to train ChemXSeer’s CRF. The second was to merge the results from all the aforementioned taggers along with ChemXSeer in a way that would improve recall, while sustaining high levels of precision.

3.2.1.1 Modified ChemXSeer

ChemXSeer utilizes two CRF modules, one for extracting formulas and another for extracting chemical names. We created a new unified CRF extractor for all *chemical entities*. The unified extractor merges features that were used for chemical formulas and chemical names. The used features include

- The word itself
- Character level n-grams
- Prefix, postfix, inclusion of punctuation, has superscript
- Regular expressions to match alphanumeric patterns such as the state of capital letters in the word (starts with, mixed caps, ends with cap), the occurrence of digits
- Dictionary look up against a collection of chemical names, chemical elements, known symbols and abbreviations

We also use a window of size n for features of the previous and following n words to be included in each word’s features, where n is set to 1 or 2 based on the feature. After tokenization with the ChemXSeer chemical tokenizer which is based on Lucene StandardAnalyzer [46], each token is assigned to one of the following classes: $\{B, I, O\}$, where B denotes a start of chemical entity, I denotes a continuation from the previous entity, and O for everything else. Three models are created for the purpose of evaluation, one is trained on the training data, the second is trained on the development dataset, and the third is trained on both training and development datasets.

After tagging the sequence of words in a document, those identified as class B or I are passed to a chemical entity parser which validates that the token is actually a chemical entity. Also we compile a list of common false positives which we denote to as a blacklist. The list can be found on code repository website. An entity candidate is ignored if it is found in the blacklist.

3.2.1.2 Ensemble extraction

We run all the three extractors, ChemxSeer, OSCAR4, and ChemSpot, on the datasets and combine their output as follows. Let token t be identified as a chemical entity by at least one of the extractors where t is defined as an offset and length only therefore it can refer to unigram or multi-gram token. Assume we have n chemical entity extractors, then we are interested in measuring the probability of t being an actual entity given the predictions from the n extractors. That is, we would like to estimate

$$P(t = Entity|E_1..E_n) \quad (3.1)$$

where E_i is an indicator random variable representing the prediction of chemical entity extractor i , $i \in \{1, n\}$, for the token t . This represents a discriminative model that tries to infer t given all the results from $E_1..E_n$. Luckily, estimating the probabilities and the final conditional probability is not hard because it follows from the performance of each extractor on the annotated dataset. In the case of a single extractor i , we can estimate $P(t|E_i)$ as the precision of extractor i on the annotated corpus. When two extractors are used, i and j , the conditional probability $P(t|E_iE_j)$ can be estimated by computing the precision resulting from intersecting the list of chemical entities identified by extractor i and j . $P(t|E_iE_j)$ is interpreted as the probability of correctly identifying a chemical entity when *both* extractors i and j identified t as a chemical entity. In other words, both extractors are used to identify chemical entities, and the intersection of their output is used to compute the precision, which is the estimated conditional probability. Finally, estimating $P(t|E_i\bar{E}_j)$, meaning that extractor i has identified t as a chemical entity while extractor j has not identified it as such, is carried by computing the precision resulting from extractor i and not j . In other words, it is the precision of an extractor whose output is given by $\{x : x \in i \wedge x \notin j\}$. This approach is generalized

to estimate the probabilities using n extractors.

For example, let CO_2 be a token that was identified by ChemSpot and OSCAR4 only where ChemXSeer failed to recognize it as a chemical entity. So $E_{chemspot} = 1$, $E_{oscar} = 1$, and $E_{chemxseer} = 0$. The confidence of the term CO_2 is given by

$$P(CO_2 | E_{chemspot} = 1, E_{oscar} = 1, E_{chemxseer} = 0) = Precision(Y)$$

$$Y = \{x : x \in chemspot \wedge x \in oscar \wedge x \notin chemxseer\}$$

So Y is an extractor whose output results from the intersection between OSCAR4 and ChemSpot, minus ChemXSeer.

Since there are 2^n possible combination for the output of extractors, we need to estimate $2^n - 1$ parameters for the probabilistic framework to output probability for every possible combination (note that we do not need to estimate the probability when none of the extractors identifies a token to be a chemical entity). While this scales exponentially, it is actually quite easy and fast to estimate the parameters because the expensive part is the extraction itself, and not merging or intersecting results. Since the extraction is done before hand, estimating the parameters only takes fraction of the time needed for extraction. In our case, we have used 3 taggers only, hence there was 7 parameters to estimate.

Ensemble information extraction has been applied before in many applications, including the CHEMDNER challenge that we participated in. One of the popular methods in using multiple extractors together is to feed the output of one extractor as an input to a second extractor to be used as a feature. This approach is often referred to in the literature to as *stacking* [47,48], where multiple extractors are stacked on top of each other such that the output of of several base learners is used as input for the following layer learner. In CHEMDNER multiple teams applied stacking by using the output of ChemSpot as a feature, with a CRF model comprising the final extractor [49–51]. In this case, stacking introduces a serious limitation as the model parameters of the CRF become highly-dependent on the output from individual extractors resulting in smaller weight being assigned to other important features [31]. Another limitation of stacking appears when the extractors use different tokenizers and do not allow tokenization to be performed outside of the classifier software package. In this case, the tokens are not the same, therefore the receiving extractor cannot benefit from the output of the preceding

extractor.

Combining the output of multiple extractors in a probabilistic way has been introduced earlier [52]. The approach used in [52] relies on linear interpolation of the classifier class probabilities. The final probability is a weighted average of the individual classifier probability multiplied by the importance of each classifier. The parameters are estimated using cross validation. When the weights are symmetric, each classifier is given equal vote, and the problem becomes majority voting by the collection of extractors. Our method, on the other hand, estimates the actual probabilities of merging the results which can use certain dependencies between the random variables. This way, probabilities depend on the combination of underlying classifiers that generated the output, and not simply on the number of classifiers that generated that output which is the case in majority voting. Furthermore, we do not assume independence of the extractors. This allows us to capture certain relations like what is the probability of an entity being a chemical entity when only ChemSpot and OSCAR4 recognized it as such, while ChemxSeer failed to? This is more powerful than relying on the conditional probability when any two extractors identified the entity as the case of majority voting.

3.2.2 ChemXSeer Tagger 2.0 - post competition

After the end of the CHEMDNER challenge, we seek to identify areas of improvements that would enhance the extractor’s performance. We start by examining the tokenization process as it is the first step in any information extraction application. We later focus on crafting new set of features that would capture the characteristics of chemical entities. These features are used to build a Conditional Random Fields extractor.

3.2.3 Tokenization

Tokenization has a significant effect on the performance of any information extraction system as tokenizers provide the tokens on which the extractor operates. However, very little attention was paid to the quality of the tokenizers in the CHEMDNER challenge. Therefore, a study of the performance of tokenizers is needed to justify using one tokenizer instead of the other. The performance of three prominent tokenizers is studied here by examining how accurately they identify

Measure	ChemSpot	OSCAR4	ChemXSeer
Correct	17149	20491	17869
Split Correct	2379	1744	3190
Total Correct	19528	22235	21059
Incorrect	5823	3116	4292
Accuracy Percentage	77.03%	87.7%	83.06%

Table 3.1. Accuracy of multiple tokenizers when tested on the chemical entities of the test set

the boundaries of the chemical entities in the test data set of the CHEMDNER task. The test set was chosen instead of the training and the development dataset because the annotations were corroborated by a second annotator.

Each tokenizer, ChemSpot, OSCAR4, and ChemXSeer is run to generate offset and length of each token in the test corpus. The results of the tokenizers are given in Table 3.1. In the table, Correct refer to the number of chemical entities in the test set that were tokenized correctly by the tokenizer. In other words, the tokenizer correctly identified the offset and the end of the token within the document in accordance to the tags provided in the test set. The *split correct* refers to the number of tokens spanning multiple words where space is the only allowed word separator, that were identified correctly by the tokenizer. Overall, there were 25,351 chemical entities in the test set. OSCAR4 had the highest accuracy rate in tokenizing chemical entities at 87%.

The tokenization accuracy on the chemical entities provide an *upper bound* for the highest possible recall by an extractor using the provided tokenizer without performing any post processing on the identified tokens. So, the highest possible recall of an extractor using OSCAR4 tokenizer would be 87%, unless this extractor uses post processing techniques. This helps in explaining the sources of error and potential areas for improvement when it comes to designing better extractors.

Since OSCAR4 tokenizer had the highest accuracy, we adapt it as the default tokenizer in the ChemXSeer Tagger 2.0. The other two tokenizers are available to the CRF extraction software, but OSCAR4 is the default tokenizer.

3.2.4 Extractor and features

ChemXSeer Tagger 2.0 uses the Conditional Random Fields implementation provided in Mallet [53] to identify chemical entities in the CHEMDNER corpus.

We train the CRF using a “first-order model”, thus each pair of labels and observations is assigned a weight. This configuration captures global state information effectively while at the same time avoid over-fitting. Limited Memory BFGS algorithm is used to train the model. Similar to the CRF model developed before the competition, *BIO* is used to label the words.

As CRF works on the sentence level to identify the true labels of the words within each sentence, *Apache openNLP* [54] is used to detect sentence boundaries. Each sentence is then tokenized using OSCAR4 tokenizer. Later, features are extracted to represent each token in the sentence. The feature classes are described below:

3.2.4.1 Word embeddings

Often in many information extraction applications, new terms will show in the test cases that have not been seen previously in the training dataset. To overcome this challenge, word embedding features are incorporated while building the model. The idea behind word embedding is to assign the words of a chosen corpus into multiple clusters such that all the words belonging to a single cluster are *related* to each other. At test time, the cluster Id of the term is used as a feature allowing the model to link the term to other terms that appeared previously in the training dataset using cluster information. Since the words in each cluster are *related*, a new unseen word that belongs to a given cluster that often contains named entities is likely to be a named entity. Word embeddings can be thought of as a transformation of the term to a finite space where elements from this space have been observed previously during the training phase.

The corpus is usually chosen to be large enough such that large number of terms will be observed. In addition, the corpus needs to be representative of the domain of the documents that contain entities which need to be extracted. For example, to extract people names and location information, a corpus about news articles can be used, while to extract chemical entities, a corpus that is built with chemical documents is needed.

Word	Cosine Similarity
Ca ₂	0.838966
Ca	0.692185
Thapsigargin	0.565048
Stores	0.562570
Potassium	0.549055
Magnesium	0.539387

Table 3.2. Similar words to calcium when sorted by cosine similarity

Term	Cluster Id
Tetralinoleoyl	8
thiophosphocholine	8
Phosphoethanolamines	8
y505f	10
Vav	10
Tsad	10

Table 3.3. Example of word embedding clusters

Many approaches have been proposed to observe word embeddings features including *Brown Clustering* [55] and *Word2Vec* [56]. In Brown Clustering unigrams are clustered hierarchically based on the bigrams in which they appear, thus forming a dendrogram that is encoded using Hoffman code. At training and testing, the Hoffman code or a prefix of the code are used as features to describe the term. In word2vec each term is transformed into a vector based on the surrounding terms appearing next to it in a predefined window. Neural nets are used to infer the vector space representation of each term. For example, Table 3.2 shows the list of words whose vector representation is similar to *calcium* when cosine similarity is used as distance measure. Later, the vector representation of each word is used to cluster the terms using K-Means. Table 3.3 lists examples from two clusters. Note that words in cluster 8 are chemical entities.

We run word2vec on a corpus containing 700,000 PubMed abstracts for articles appearing in journals where the journal name had the word *chemistry* in it. The corpus contains 213,030 unique terms, that appear 143,301,537 times. We use the default parameter values of word2vec and choose the number of clusters to be 1000. A hash map between the terms and the cluster Ids is created to be used in training

Soundex Code	Letters
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R
No Code	A, E, I, O, U, H, W, Y

Table 3.4. Soundex code for English letters

and evaluation. As part of the feature generation, each term will be looked up in the hash map for the value of the cluster Id. If the term has not been seen in the corpus, i.e. does not have a cluster Id, the feature is not set.

3.2.4.2 Soundex features

Soundex is an algorithm that is used by the USA Census Bureau to encode surnames *phonetically*. The generated code contains the first letter of the surname combined with three digits representing the the last name *phonetically* (how do they *sound*). Each digit represents a collection of letters that are phonetically similar. Table 3.4 shows the letter mapping that is used by the Soundex algorithm implemented at the Census office [57]. If the word contains more than three encodable letters , which is the default Soundex implementation, the remaining letters are ignored.

Soundex is especially effective in matching names with multiple spellings and overcoming spelling mistakes. Soundex provides a powerful mechanism for matching homophones (words that are pronounced similarly but are written differently). For this reason, we borrow this technique as many chemical names tend *sound* similar, albeit being spelled and structured differently. For example, *carbon*, *carbonate*, *carbonic*, and *carbonyl* all have the same Soundex code **C-615**. This transformation helps the extractor in identifying chemical entities that did not appear in the training set, but a phonetically similar entity appeared.

We are not aware of any work that utilizes *Soundex* code to identify chemical entities. In our extractor we use the implementation of *Apache openNLP*, while we set the maximum number of allowed digits at 7. Thus we allow the algorithm to encode more letters than the typical Soundex implementation. At training and

evaluation, each term is converted to its Soundex code, and a feature is set for each unique value of the Soundex code.

3.2.4.3 General token derived features

A collection of features are derived from the term itself and its shape. These are the following:

- The word itself and lower case version of the word
- Regular expressions to identify if the term contains digits, starts with a capital letter, all capped, all small, mixed cap and small, ends in a sign, ends in a number, contains dash, starts with a number
- Character level n-grams of length 2, 3, and 4
- Heuristic to identify formulas by setting a feature when no two consecutive characters in the term are small case
- Selected features from neighboring terms
- Whether the term marks beginning of a sentence. This is useful in distinguishing proverbs that are capitalized at the sentence beginning from others that are intentionally capitalized
- NLP features based on *BANNER* [58] including lemmatization, and word class conversion

3.2.4.4 Dictionary look up

Dictionaries are used to generate features corresponding to the existence of a given term, or part of it, in the dictionary. The main dictionary used was *Jochem* [59] which contains more than 1.6 million chemical names that were captured from multiple databases. The chemical entities in Jochem were tokenized using OSCAR4 tokenizer because many of them are multi-term entities that would not match a unigram token, which is the unit of tagging in the CRF model. Dictionary tokenization is necessary to ensure that dictionary look up is effective and to avoid the need for prefix matching with a huge dictionary like that of Jochem.

Beyond Jochem, smaller dictionaries were compiled to capture amino acids and their abbreviations. Another dictionary was used to match common prefixes and postfixes that appear in specific groups of chemicals like organic chemistry. A special dictionary of *boost terms* was used to boost certain terms that were occasionally missed by the extractor. In this extractor, the black list has been dropped.

3.3 Results

In this section the results are presented and discussed for the classifier used during the competition, and the CRF classifier developed after the competition.

3.3.1 Competition extractor - ensemble approach

The performance of the ensemble extractor is presented on both the CEM and CDI tasks. In the ensemble approach, OSCAR4 was combined with the output of ChemSpot and a modified version of ChemXSeer. While ChemSpot and OSCAR4 were used out of the box and did not make use of the provided training dataset, ChemXSeer was trained and tested on opposite datasets. That is, to test ChemXSeer on the development dataset, the model was built using the training dataset only. The parameters of Equation 1 were found to be close enough whether estimated using training or development datasets. Therefore, the final test dataset used the development estimate probabilities. Table 3.5 shows the estimated probabilities when conditioned on all the possible values for the extractors outcome.

ChemxSeer	OSCAR4	ChemSpot	Probability Estimate on Dev	Probability Estimate on Train
1	0	0	0.252	0.26159
0	1	0	0.089	0.08507
0	0	1	0.249	0.25588
1	1	0	0.82083	0.81755
1	0	1	0.72799	0.67361
0	1	1	0.55869	0.53267
1	1	1	0.93316	0.93386

Table 3.5. Probability of a candidate entity conditioned on possible values of the indicator random variables for each of the three taggers used.

Dataset	Threshold	Precision	Recall	F-Measure
Dev	0.01	0.31543	0.8924	0.46611
Dev	0.24	0.67406	0.73650	0.70390
Dev	0.25	0.70871	0.71598	0.71232
Dev	0.5	0.79486	0.67544	0.7303
Dev	0.7	0.87369	0.55663	0.6800
Dev	0.8	0.88315	0.52835	0.66116
Dev	0.9	0.93316	0.30973	0.46509
Train	0.01	0.30711	0.88147	0.45552
Train	0.25	0.66208	0.73126	0.69495
Train	0.26	0.78473	0.66680	0.72098
Train	0.5	0.78473	0.6668	0.72098
Train	0.6	0.86928	0.55312	0.67607
Train	0.7	0.88266	0.52568	0.65893
Train	0.9	0.93386	0.31135	0.467

Table 3.6. Performance of the ensemble extractor on the CEM task at various confidence thresholds.

Using the probabilities generated from our probabilistic framework, we can apply cut off points based on the confidence assigned to each extracted entity. We have experimented with multiple thresholds and found out that at low threshold values, the recall is favored. The precision is favored over recall as the threshold value increases. Some of the obtained results for the CEM task are summarized in Table 3.6. The highest obtained F-measure was 73% on the development data, and 72% on training data. Similarly the recall reached a maximum of 89% for development, and 88% for training. Interestingly, one will be able to obtain near 73% recall at 66% of precision. That is, we are able to identify nearly 3/4 of all chemical entities in a document with only 1/3 of these identified entities being false positives. In Figures 3.1 and 3.2, we plot precision against various values of recall for the CEM task using both training and development datasets.

3.3.2 Post competition - ChemXSeer Tagger 2.0

The single CRF extractor with the novel feature set was trained using both the training and the development dataset, and tested on the test set. The test was done on the CEM task only, as the extractor was optimized for this task rather than CDI. Multiple runs with different collection of feature set were conducted, and the

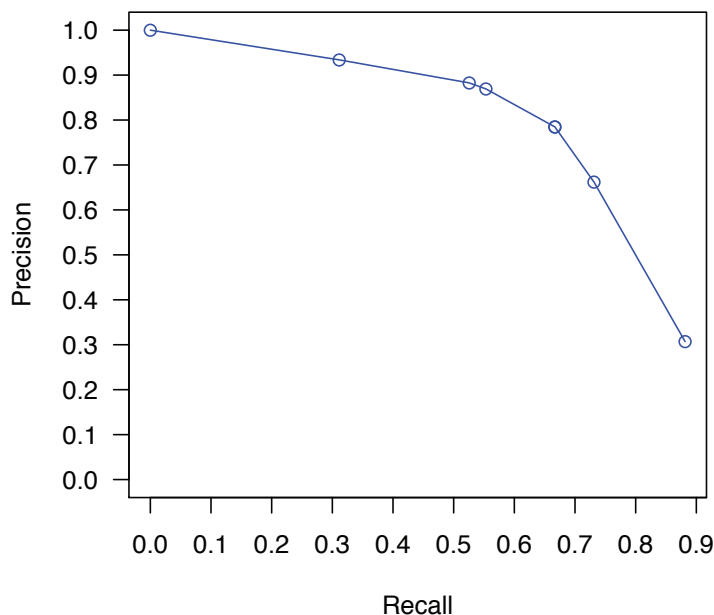


Figure 3.1. Precision-Recall curves for CEM task on training dataset

Run	Precision	Recall	F Score
ChemxSeer Tagger 2.0	0.89569	0.78009	0.83390

Table 3.7. Performance of ChemXSeer 2.0 Tagger on the test dataset

best performance was obtained using the combination of all the features. The result is shown in Table 3.7. The F1 score was 83.3%, a significant improvement over previous standalone extractor performance. It is worth mentioning that ChemXSeer Tagger 2.0 does not perform any post processing or abbreviation matching, despite the existence of abbreviations in the dataset. That is because abbreviation matching can be performed by third party tools without the need to complicate the code base.

The effectiveness of the CRF extractor when multiple set of features are used is shown in Table 3.8. When word2vec features are removed, the extractor’s accuracy drops the most. Soundex features have relatively small contribution in the presence of word2vec features. However, varying the Soundex code length have an effect

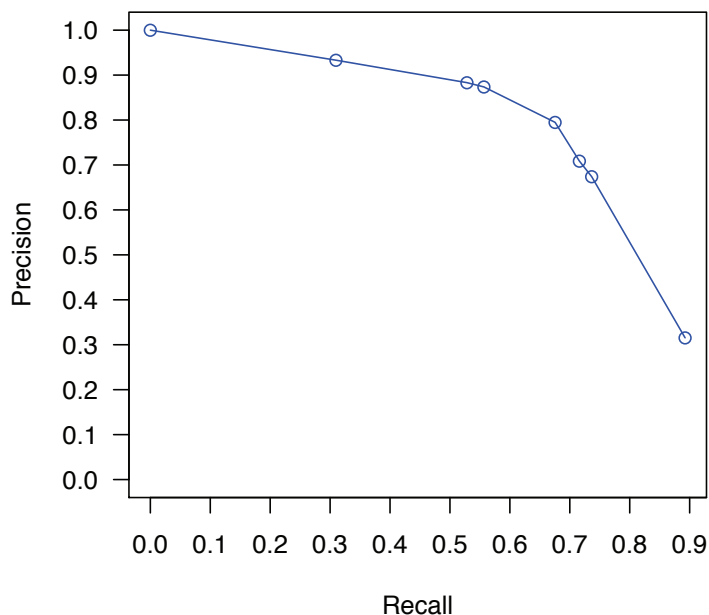


Figure 3.2. Precision-Recall curves for CEM task on development dataset

Feature	Precision	Recall	F Score
All	0.89730	0.77646	0.83252
All - NLP	0.89749	0.74151	0.81208
All - Word2Vec	0.88993	0.75393	0.81631
All - Soundex	0.89784	0.77342	0.83100
Soundex 3	0.88937	0.76869	0.82464
Soundex 5	0.89647	0.77606	0.83193
Soundex 7	0.89730	0.77646	0.83252
Soundex 100	0.88868	0.76995	0.82507

Table 3.8. Performance of ChemXSeer 2.0 Tagger using different collection of features

on the extractor’s accuracy as a very short code is not powerful in discriminating chemical entities while a very long code, 100, is counter productive.

ChemxSeer Tagger 2.0 has room for improvement. By recalling that the tokenizer had an accuracy of 87%, which is an upper bound on the attainable recall, and comparing that with the 78% recall that ChemxSeer Tagger 2.0 achieved, the tagger

can still enhance its performance significantly by identifying the missing 10% of the total entities.

3.4 Conclusion

We introduced an ensemble approach for chemical entity recognition that employs multiple extractors and output probabilities that represent the confidence score for each entity. We showed how these probabilities can be estimated using the training dataset in an effective way. In implementing this approach, we use a modified version of ChemXSeer along with ChemSpot and OSCAR4. Our approach generates probability values that can be used for thresholding each prediction. This aspect can be used to trade-off precision vs. recall. With a higher threshold for probability, our method extracts highly-accurate entities whereas for optimizing recall, a lower threshold on probability can be enforced.

We have also conducted a study about the accuracy of chemical text tokenizers where it was found that OSCAR4 tokenizer outperform others on the CHEMDNER test set. A new extractor, ChemXSeer Tagger 2.0, is built as a CRF extractor that utilizes OSCAR4 tokenizer. We introduced a set of novel features, including word embedding and Soundex that are used in building the CRF extractor which achieves 83.3% F1 score on the test set without any post processing or abbreviation matching. ChemXSeer Tagger 2.0 is designed as an API and can be used as stand alone program. The source code is available on Github: <https://github.com/SeerLabs/chemxseer-tagger>

Chapter 4 |

CiteSeerX Log Mining

With academics and scholars relying more on digital libraries available on the web to obtain copies of research articles than waiting for the printed journals and proceedings to arrive, it is of interest to examine how scholars access these articles by studying their access logs for such digital libraries. In this chapter we perform a large scale study on a public digital library's access logs for a period of three years, within which more than 3 billion records have been accumulated. More than 200 million of these records are identified to originate from human users and not search engine bots. The access patterns of users are studied along with the papers they are interested in and their downloading behavior. We model the number of downloads a paper receives and show that a power law distribution is a good approximation. Furthermore, analysis shows that the top downloaded papers show a repetitive pattern. As such, we try to predict the popular papers for a given month based on the access behavior from the previous months. Additionally, analysis of the queries issued by users to the digital library's search engine indicate a significant interest in searching for author names more than those with typical full text queries.

4.1 Introduction

As digital libraries available on the web such as ACM DL and IEEE Xplore along with online versions of many journals become increasingly popular, it is natural to study the characteristics and interests of the users based on their interactions with the websites and the implicit feedback left in the form of access logs. This is further supported by the shift of users towards obtaining digital copies of research articles more than print versions [60–62]. Examining access logs can be used to understand

the user experience with the digital library, track down paper level of interest, and later use such information to provide a better service to the users. Analyzing access logs has a long history, e.g. to be used as an implicit feedback tool to learn ranking functions of search engines [63], or measure affinity between a social network of friends [64]. Here, we aim to uncover the characteristics of user interactions with a publicly accessible digital library using the logs from September 2009 to March 2013. During this period, the library’s web servers have collected more than 3 billion entries with nearly 133 million of them being download attempts. To the best of our knowledge, this is the largest study of digital library’s access logs in terms of volume and duration.

The access logs of CiteSeerX digital library and search engine are used in this work. CiteSeerX contains more than 5 million scientific documents, including papers, dissertations, books, and technical reports in the area of computer and information science with the recent addition of physics, economics, medicine and other areas. The metadata of these articles is extracted automatically including authors, title, and citations. CiteSeerX receives more than 2 million requests per day from more than 200 countries, making it one of the most popular repositories in the world¹. We focus on the paper download behavior and queries issued to CiteSeerX search engine. These two tasks are the most common interactions a user would have with a digital library. Furthermore, none of them requires registering on the site which unlike making corrections to the metadata would require logging in with an active account. Therefore these two activities capture the behavior of a wide variety of users.

Upon examining the top downloaded papers in each month it was found that the top 1% most downloaded papers tend to show up again in the top downloads later. Given that the top 1% most downloaded papers account for 10 to 20% of the total download requests received by CiteSeerX from users identified as *humans* and not bots, we investigate the ability to predict the top downloaded articles of the upcoming month. Indeed predicting the most downloaded articles is beneficial for the users and the system, since the system can use a caching server to store the predicted highly accessed papers which in turn would reduce the latency for the user, and reduce the workload on the servers.

The second most frequent interaction users have with a digital library is discov-

¹<http://repositories.webometrics.info/en/world>

ering and searching for articles and author profiles. CiteSeerX is unique in that sense as it provides many items that can be searched within a research document such as tables and algorithms. In addition, advanced search can be used to query certain fields, such as the year in which the article was published, title based search, and abstract search. With CiteSeerX’s indices being searched 8 million times, by *human* users, in the period of the study, we were able to study the distribution of queries sent by users. The search behavior combined with the click through logs can be used to learn ranking functions that better serve the user queries by returning more relevant documents first.

4.2 Data Processing

The data with which this work was carried consist of the web access logs of CiteSeerX web servers. In production, CiteSeerX uses two web servers that are behind a load balancer. The web servers run Apache Tomcat 6 and write rotating logs in the combined log format². The stored information include: the client’s IP address, the client’s name, the time of the request, the requested URL, the referrer URL, the user agent, the response code, the amount of bytes sent back, and the session identifier. The logs generated during the months of September 2009 to March 2013 were used in this study, totaling 100 GB of compressed files. The log files are processed using *Pig* scripts [65, 66], and each of the above mentioned fields is extracted before being imported into a Hive table [67, 68]. The number of imported hive entries came to 3,317,634,711.

Studying the logs at the session granularity is important to understand the context in which interactions with the site are taking place. We apply an automatic session detection heuristic to cluster the different requests from the same IP address into multiple sessions. We use the approach proposed by Radlinks and Joachims [69], which later became an established approach [70] in the industry. In this method, all requests from the same IP are grouped together and sorted in chronological order. Then, consecutive requests with no more than 30 minutes of idle time between any two consecutive requests are assigned to the same session. That is, whenever 30 minutes pass without receiving a request from the IP address, the next received request will start a new session. The total number of sessions in the logs was

²<http://httpd.apache.org/docs/1.3/logs.html#combined>

241,777,051.

Since we are mainly interested in the behavior of real users and not search engine bots, we have to exclude sessions generated by bots. We obtain a list³ containing 16988 IP addresses along with user agent information of known search engine bots. This list was used to filter out sessions coming from IPs in this list or having a user agent that was tied to a bot IP in the same list. While this list may not be complete, it is the largest we were able to obtain. We further filtered out any session in which the user agent contained any of the words *crawler*, *bot*, or *spider*. The last step targets crawlers that are not widely known, but take the courtesy to identify themselves as such. However, these steps are not enough to detect bots masquerading as human users, therefore we use a heuristic approach similar to that of Tanasa [71] after augmenting it with more rules. Basically, a session S is considered to be from a bot, and not a human user if one of the followings are true:

- There is a request in S that used ‘Head’ request method
- The session S requested */robots.txt*
- The session reached CiteSeerX’s page that is shown when the daily download limit is exceeded
- The session S contains at least θ_1 requests, with $k_1\%$ of them sharing the same referrer.
- The session S contains at least θ_2 requests, with $k_2\%$ of them sharing the same referrer.
- The session S contains at least θ_3 requests, with more than k_3 request received per second.
- The session S contains at least θ_4 requests, with more than k_4 request received per second.
- The requests within the session had more than one unique user agent. This can be an indication of proxy servers. While it might not be necessarily a bot session, it is however a mix of multiple sessions.

³http://user-agent-string.info/rpc/get_data.php?botIP-All=csv

- The session S contains at least θ_1 requests, with $k_5\%$ of them being referred from DBLP.
- The session S contains at least θ_1 requests with the requests being an ordered enumeration of paper Ids in CiteSeerX. For example, requests to download papers 5, 6, 7, 8, 9, ...,100.
- The session S contains at least θ_1 requests with the macro average of number of requests per second, only for the seconds at which requests were received, is larger than m
- The session S contains at least θ_1 requests with at least $n\%$ of them returning a 302 response code. This is an indication of an attack that is automatically trying to generate URLs that do not exist.
- The session S contains at least θ_1 requests with the most common timespan between two consecutive requests in the session happening at least $w\%$ of the times. That rule identifies robots that idle for a fixed amount of time before sending a request. If this idle time is too frequent, the session is flagged to be that of a bot.

In the above rules, the following values were chosen for the parameters. $\theta_1 = 100$, $k_1 = 80$, $\theta_2 = 1000$, $k_2 = 30$, $\theta_3 = 20$, $k_3 = 1$, $k_4 = 0.25$, $k_5 = 50$, $m = 3$, $n = 25$, and $w = 30$.

Also excluded are the static media files such as javascript, icons, and images. This resulted in keeping 203,505,481 log entries out of the 3,317,634,711 total.

After generating sessions IDs and filtering out bot requests we create a hive table for paper downloads, which contains the DOI (digital object identifier) of the downloaded paper, the time of request, and the IP address downloading the paper. This information was extracted from the processed logs using a regular expression matching requests to the download handler. Any download request that returned a response code other than 200 was filtered out. This is important because CiteSeerX imposes a daily limit on the number of allowed downloads per IP; beyond this limit requests are redirected to an error message page. Similarly, search requests were identified using regular expressions that matched the search query handler in CiteSeerX. The query text along with timestamp, session, the order of results page, the client's IP address as well as the search type were extracted and stored

in a Hive table. There are three types of search that are supported directly by CiteSeerX: *Document Search*, *Author Search*, and *Table Search* with *Document Search* being the default type of search when no type is specified.

4.3 Access Analysis

In this section we report the results of our analysis on multiple fields. Note that only requests identified to originate from humans are used in this analysis, unless otherwise stated.

4.3.1 Session

We begin by examining statistics of the session information. The average length of the session is 1.85 requests, with the minimum length being 1 request and the maximum being 2447 requests. These numbers indicate that most visits to CiteSeerX digital library are approximately of length 2.

In Figure 4.1 the length of the session in terms of the number of requests is plotted against the frequency of this length where both axis are in log scale. The figure resembles that of a power law distribution, albeit with more distortions towards the tail. Therefore we try to fit a power law distribution using Clauset method [72]. In a power law distribution the probability of observing a given quantity, x , is given by $Pr(x) \propto x^{-\gamma}$ where γ is constant. Here x refers to the length of a given session in terms of number of requests. The estimated γ parameter of the power law distribution was 1.391 with p-value of 0.99 using the Kolmogorv-Smirnov goodness of fit test, when the minimum x value is 6542. This means that the power law distribution is a very good fit for the empirical distribution. It is worth mentioning that real world observation rarely follows power law distribution, and in most cases they can only be approximated with a power law towards the tail of the distribution [72].

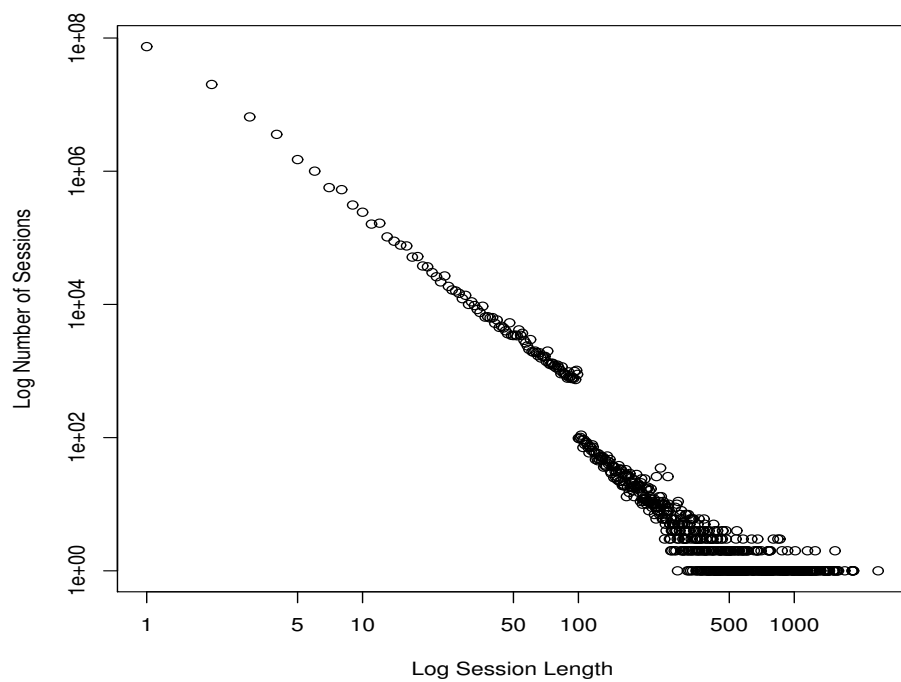


Figure 4.1. Plot of the length of the the session (number of requests in the session) against the frequency of this length in log log scale.

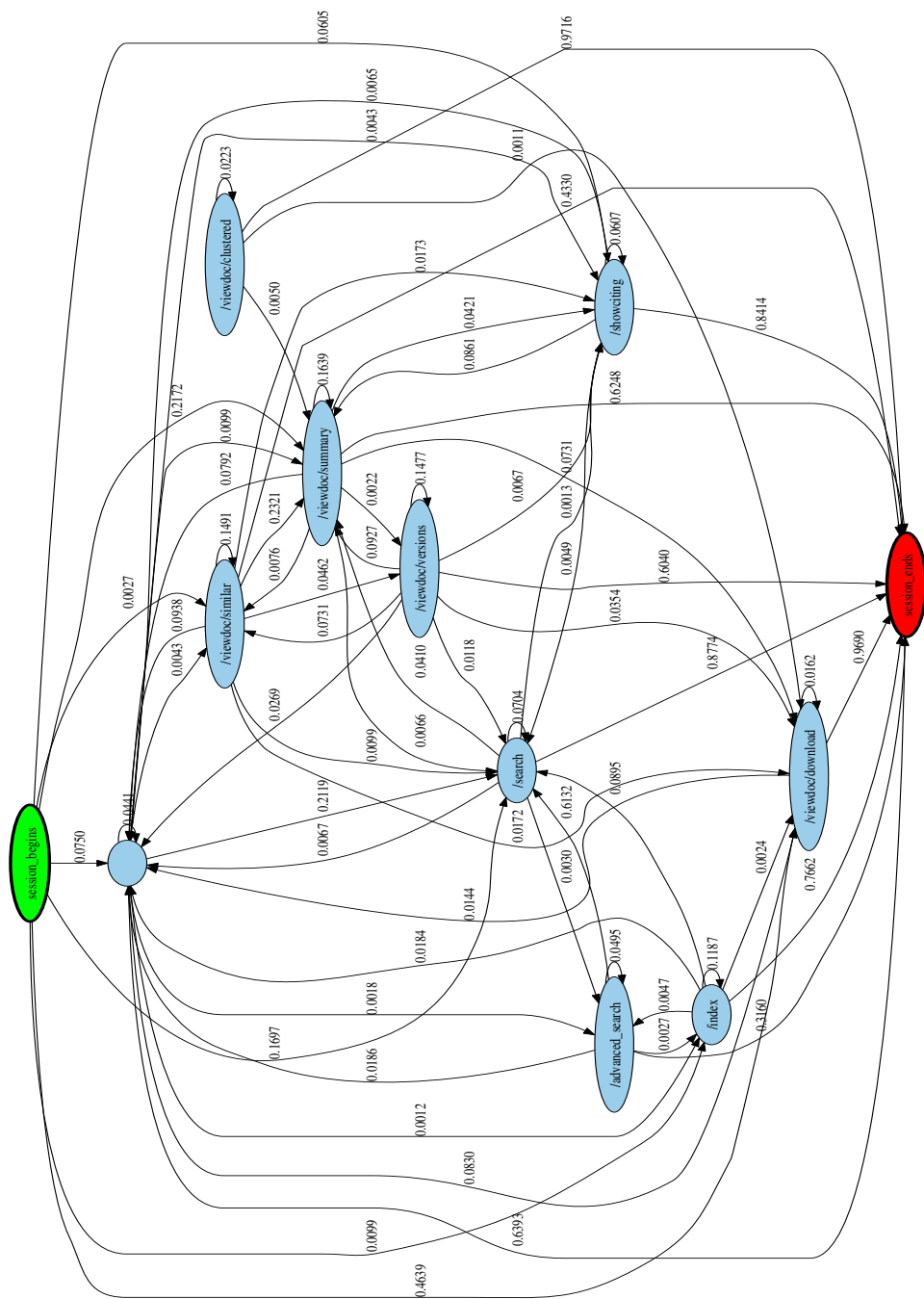


Figure 4.2. A state transition graph of web pages covering searching, viewing and downloading documents

It is often informative to examine the transition behavior between different web pages in the same session as it helps explaining where most of the users start their session and where do they *bounce* off the site. Therefore, the transition behavior between a selected group of pages is examined in order to build a state transition diagram. To build the diagram we first cluster the requests based on a predefined collection of URL prefixes such that each prefix matches a certain task a user would take like download, search, advanced search, look at similar documents ..etc. For each of the clusters we create a state in a state diagram. We then create a directed edge between two states, from a to b , representing that a user transits from a to b with probability estimated as the proportion of times users move from a to b compared to navigating from a to all states. The state transition graph for search, downloads, and viewing documents is given in Figure 4.2. Interestingly, we find that 46% of the sessions starts at a download page which they arrived to from an outside referral, and 7% of the sessions began at the main page indicating that more users arrive at the site to obtain a copy of a paper than any other task. On the other hand, 96% of the sessions ended after the user downloads a paper.

With CiteSeerX being an internationally known digital library and repository, it receives requests from more than 200 countries world wide. Table 4.1 lists the top 10 countries from which requests originate along with the percentage of traffic from that country. The top 10 countries account for more than 70% of CiteSeerX traffic, and 64% of download requests. The top sources of traffic in order are Google, direct traffic, Baidu, Bing, DBLP, Yahoo, Wikipedia, ScientificCommons, and Google Scholar with Google accounting for 69% of the traffic. In addition, the table shows the top 10 countries when sorted by the number of downloads received from that country.

4.3.2 Downloads

The download behavior analysis started by examining the number of downloads each paper received during the time of the study. In Figure 4.3 the number of downloads a paper receive is plotted against the number of papers with this many downloads in log-log scale. A power law distribution is fit to the distribution of number of downloads using Clauset method again. The γ parameter is estimated to be 1.69. The p-value from the Kolmogorov-Smirnov goodness of fit test is

Table 4.1. Percentage of traffic from the top 10 countries

Country	Traffic %	Country	Download %
USA	24.31%	USA	23.99%
India	8.83%	India	10.58%
China	7.63%	China	6.13%
UK	5.44%	UK	5.87%
Germany	4.81%	Germany	3.89%
Canada	2.61%	Canada	2.75%
France	2.51%	Iran	2.22%
Iran	1.96%	France	2.03%
Japan	1.79%	Australia	1.98%
Australia	1.74%	Korea	1.74%
Other	38.33%	Other	38.76%

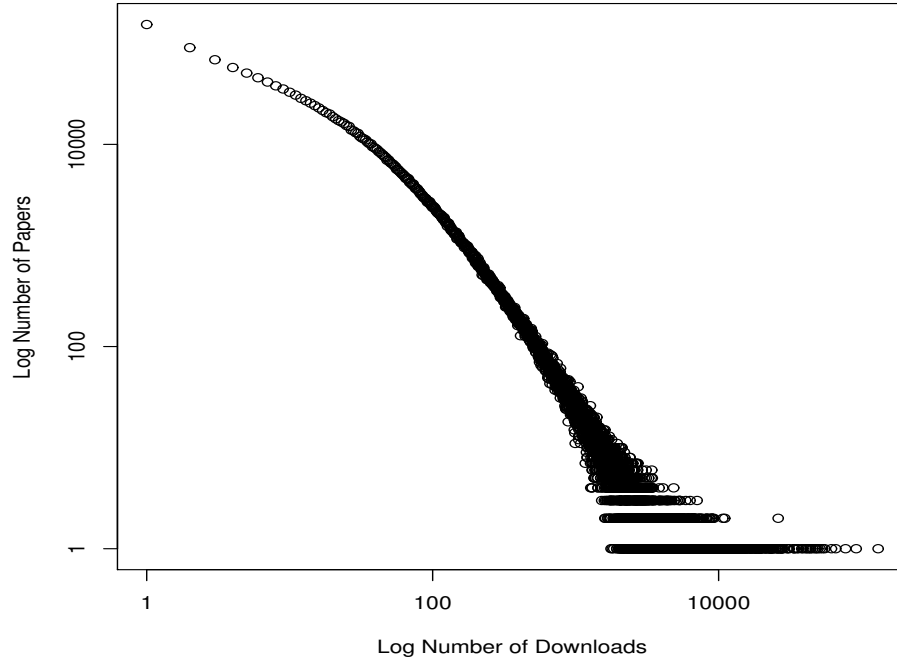


Figure 4.3. Plot of the number of downloads a paper receives against the number of papers with this many downloads in log log scale.

0.41 which is achieved when x is limited to be above 820. This indicates that a power law distribution is good in modeling the number downloads a paper receives. Again, we reiterate that real world observation rarely follows power law distribution, and in most cases they can only be approximated with a power law towards the tail of the distribution [72]. This is the first work to analyze the distribution of number of downloads a paper receives, opening up the door for future work in this area, therefore a power law distribution is the first step towards understanding the popularity of paper downloads. The obtained exponent for modeling the download behavior is far from the exponent obtained when CiteSeer citation networks is modeled as power law [73]. The exponent obtained for the citation data was 2.75 while the estimated exponent for the download data is 1.69.

Given that both power law distribution is used to fit both download behavior and citation data, it motivates the study of correlation between the number of times the paper is downloaded and the number of citations it has in CiteSeerX. The Pearson Correlation Coefficient is estimated to be 0.1, indicating a small positive correlation between the number of downloads and the number of citations. On the other hand, the correlation becomes 0.16 when taking the log of both number of downloads and number of citations, which helps in reducing the variance. However, we suspect that the true correlation value is higher as when we examined the papers receiving high number of downloads yet having zero citations, it turned out that in many of them the automatic metadata extractor of CiteSeerX had mistakes extracting author and title information which led to mis-assigning citations to these papers. Upon checking the citation count in Google Scholar for these papers, it was found to be significantly higher.

In Table 4.2 the top 10 most downloaded venues are listed. The first row indicates a missing venue name for a given paper, while the third venue “*In*” stems from an extraction error of the automatic metadata extractor. Both values are left to exhibit the limitations and challenges faced by digital libraries that automatically extracts paper metadata.

An interesting observation that would motivate Section 4 of this paper is drawn by computing the percentage of download requests generated by papers in the top $k\%$ most downloaded list. For $k = 1$, we see in Figure 4.4 that for most months the top 1% most downloaded papers are responsible for 10 to 20% of the total download requests. For $k = 5$ this number goes up to more than 40%.

Table 4.2. Venues ranked by the number of downloads their papers received

Venue	Number of downloads
	90,783,801
IEEE Transactions on Pattern Analysis and Machine Intelligence	234,273
In	215,248
Journal of Personality and Social Psychology	134,582
ACM Computing Surveys	129,214
Psychological Review	115,169
Communications of the ACM	113,355
Journal of Machine Learning Research	104,991
Journal of Finance	101,884
IEEE Trans. Inform. Theory	92,130
Proceedings of the IEEE	88,768

Table 4.3. Search type distribution. Blank field falls back to Document search

Search Type	Number	Percentage
Document	7,488,321	92.73%
Author	563,801	6.9%
Table	22,845	0.2%
Total	8,074,967	100%

4.3.3 Search

CiteSeerX offers three types of search where users can perform *document search*, *author search*, and *table search*. Each of these types of search is powered by a separate index. In addition, an advanced search page is available on the document index to complement the single query box default search. During the period of this study, CiteSeerX received 8,074,967 search requests. The proportion of search types are presented in Table 4.3. As seen in the table, there is a reasonable interest in searching for author names with 6.9% of the search requests targeting the authors index. It was reported in [74] that author queries account for 36% of query categories in Pubmed, however. We conjecture two possible reasons for this. In the first scenario users are interested in searching for papers by author name assuming they remember one of the authors of a paper but do not recall the title. The second reason is that individual authors are searching for themselves to track

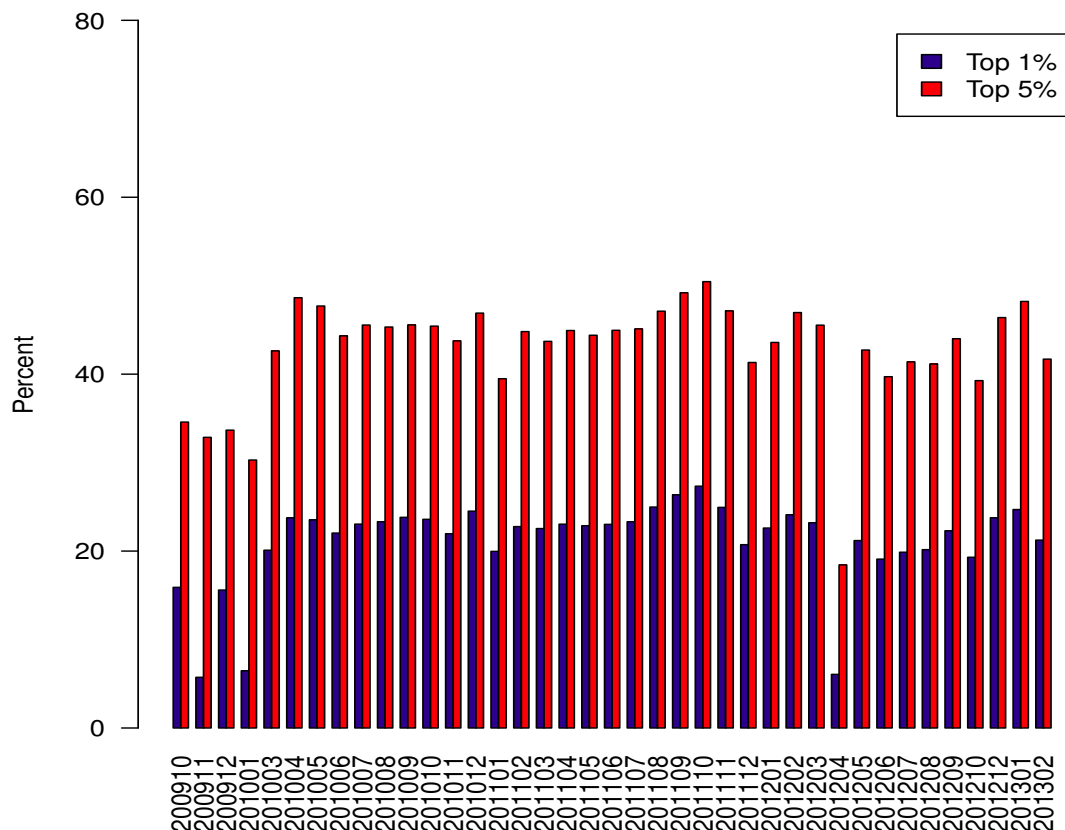


Figure 4.4. The percentage of total downloads generated by papers in the top 1% and 5% respectively of most downloaded papers within a month

down the number of citations they received, the number of papers they have in the repository, or correct any metadata that was incorrectly extracted. Similarly, tenure and promotion committees would use the author search to track down the body of work for a given scholar. Validating these assumptions and exploring the real motivation is left for future work. We also notice that advanced search was a popular choice, with that 24% of the *document search* using the advanced search box.

By examining the queries with type *document search* we found that the average length of a query is 4.76 terms. This is significantly higher than average Web search query length of 2.35 terms [75] found in AltaVista and similarly in Excite [76].

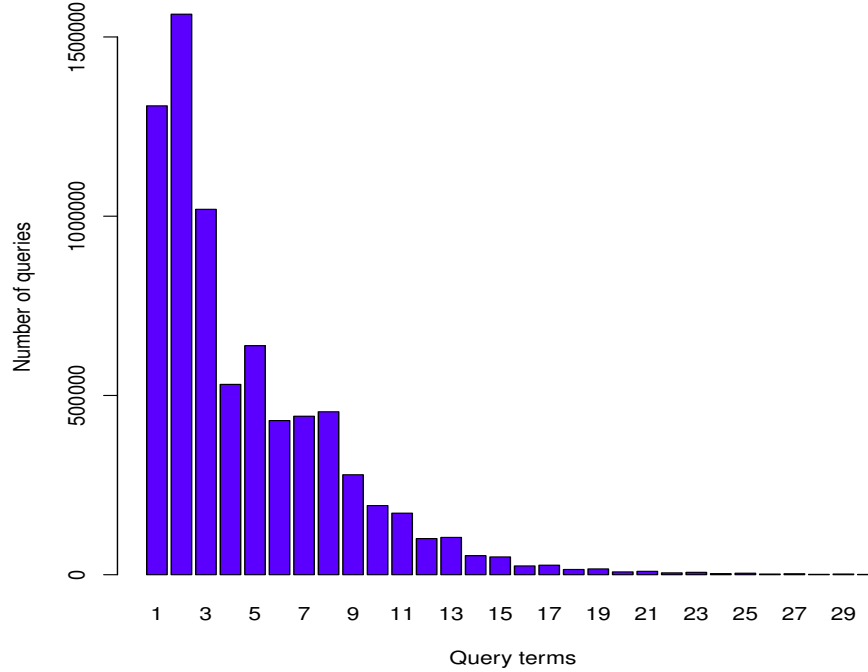


Figure 4.5. Distribution of the number of terms in a query.

However more recent studies of Web search query length put the average at 2.9 [77] and 3.08 [78]. It is, therefore, believed that the number of query terms has increased over time [78], which explains the different in the number of query terms we found with that found in the Elsevier’s ScienceDirect OnSite query logs where the reported average query length was 2.27 [79]. Figure 4.5 shows a bar plot of the number of search requests received with a particular number of query terms up to 30 terms.

Majority of the users, 91.57% of them, looked at the first result page only, however users were more likely to look at 10 results pages than looking at 7, 8, or 9 pages as shown in Table 4.4. This indicates that users either found what they want on the first few result pages (or did not find it and gave up), or they were willing to explore more results to find a satisfactory result. On the other hand, in the case of of *author search*, 99% of the users browsed one result page only.

Table 4.4. The number of result pages a user looks at when doing document search

Number of Pages	Percentage
1	91.57%
2	3.57%
3	2.18%
4	0.7%
5	0.5%
6	0.2%
10	0.2%

Table 4.5. Number of document views per query

#views	ratio	cumulative ratio
1	71.14%	71.14%
2	14.94%	86.08%
3	6.02%	92.1%
4	2.88%	94.97%
5	1.56%	96.53%
6	0.97%	97.5%
7	0.59%	98.1%
8	0.4%	98.5%
9	0.3%	98.8%
10	0.2%	99%
> 10	0.8%	100.00%

4.3.3.1 Number of document views per query

We study the number of document views after submitting a query. We ignore the sessions where users issue a query but do not click on any returns. The results are presented in Table 4.5: most users click very few documents from the returned list. This suggests that most users follow the *Cascade Model* – a popular model where users view search results from top to bottom and leave as soon as they find a document of interest [80]. The multiple browsing models [81], i.e., the models that assume an query will lead to more clicks, play minor roles on CiteSeerX. The mean number of clicks was 1.36, and the maximum was 796. We suspect that the few large view counts come from crawlers that masqueraded real users.

4.4 User Behavior Prediction

4.4.1 Download Prediction

This section studies whether it is possible to predict the number of downloads each paper would receive using its download activity in the previous months. For if this is possible, the top to-be downloaded papers can be cached on a specific server that holds the files in memory, leading to a reduction in the request turn around time, and burden reduction on the web servers. However, the number of downloads a given paper receives varies from month to month, which makes it harder to predict the actual number. Instead, we focus on a similar problem, yet simpler, that is to identify the top 1% most downloaded papers. This problem is easier than predicting exact counts as it is not concerned with predicting an exact value or predicting the ordering. Nevertheless, from a practical standpoint it achieves the same goals that are being pursued by the count prediction problem; that is caching documents and predicting relative popularity.

Figure 4.4 shows that the papers in the top 1% most downloaded papers within a month account for more than 10 to 20% of the total download volume (except for three months). The top 5% most downloaded papers account for more than 30% of the total download traffic. However, we only try to predict the top 1%, because caching the top 1% in memory is feasible, assuming that on average each document is 1 MB. On the other hand, if we are to consider the top 5%, the number of papers in this category can be as high as 47,000, which can not be cached in memory using average server hardware with 32 GB of memory.

We study the potential of caching as follows: for an upcoming month x , we would like to build a caching service containing the papers that will be in the top 1% in that month. The candidate list for inclusion in the cache are the papers in the top $y\%$ of the previous k months. For example, assume $x = 2013/11$, $k = 1$, and $y = 5$ then the cache will choose from the papers in the top 5% of 2013/10. For the remainder of this work, we use $k = 1$ and $y = 5$ as they were enough to predict downloads with high accuracy.

A logistic regression model is built to predict the top 1% most downloaded papers. Logistic regression was chosen after experimenting with multiple classifiers including Support Vector Machines, decision trees, and naive bayes. In the logistic

regression model, the probability of a paper X_i belonging to the top 1% is given by:

$$P(X_i) = \frac{e^{X_i B}}{e^{X_i B} + 1}$$

The weights vector B is learned using Quasi-Newton method [82] to assign weights to the paper features. The following features are used in representing a given paper, X_i .

- *count_in_top_1_previous_months*: The number of times this paper appeared in the top 1% in the previous 12 months
- *count_in_top_5_previous_months*: The number of times this paper appeared in the top 5% in the previous 12 months
- *is_in_top_1_at_previous_month*: Was the paper in the top 1% in the previous month
- *is_in_top_5_at_previous_month*: Was the paper in the top 5% in the previous month
- *average_previous_rank*: The average rank of this paper when sorted by number of downloads in the previous 12 months
- *rank_at_last_month*: The rank of the paper when sorted by number of downloads in the previous month
- *rank_at_last_year*: The rank of the paper in the same month last year
- *num_citations*: The number of citations this paper received
- *highest_hindex*: The highest h-index⁴ of the authors
- *avg_hindex*: The average h-index of the authors

Assuming that in each month the prediction model can be learned again from the latest data, we test the approach by building a model at the end of each month, then test it on the following month. Let M_i be a given month, where M_{i+1} and M_{i-1}

⁴h-index for an author is the highest number K such that K of his papers received at least K citations

are the following and preceding month, respectively. Let $T_i(k)$ be the list of the top $k\%$ most downloaded papers in month M_i . Then, at the end of month M_i , we can compute $T_i(1)$, the list of top 1% most downloaded papers in M_i . Similarly, we have already recorded $T_{i-1}(5)$ from the previous month, M_{i-1} . Hence, the positive examples of the training data are those appearing in

$$\{\forall x, x \in T_{i-1}(5) \cap T_i(1)\}$$

while the negative training examples are the papers in

$$\{\forall x, x \in T_{i-1}(5) \wedge x \notin T_i(1)\}$$

The testing data set can be built similarly but using months M_{i+1} and M_i . The logistic regression model is built using the training data.

To compare the performance of the logistic regression classifier we consider the following baseline: for month M_i , the baseline consists of the top K most downloaded papers in month M_{i-1} , where K is the number of papers identified to be in the top 1% using the logistic regression model. Indeed, this is a reasonable baseline because the coefficient of the *is_in_top_1_at_previous_month* parameter had the highest positive value among all other coefficients. Since our candidate list was limited to the articles appearing in the top 5% in the previous month, we define an *Oracle* classifier which knows the correct classification for each paper in the candidate set. The measure by which we compare the performance of the logistic regression (LR) classifier and the baseline (BL) is *hit rate*. Hit rate is the percentage of articles in the top 1% in a given month that were found in the caching server. Therefore, higher hit rates are desirable.

In Figure 4.6 we plot the hit rate of the logistic regression classifier compared with that of the baseline and the Oracle. The hit rate of the model is also provided in Table 4.6 As shown in the figure and the table, the logistic regression model outperforms the baseline in 13 out of the 17 months. While the difference between the baseline and the model might not look significant, it is actually significant once you realize that the absolute numbers are in the thousands. Furthermore, because these cash hits for the top 1% most downloaded papers, they actually contribute to more than 10% of the overall traffic, as shown before. Hence, the slightest increase in the hit rate would translate to thousands of requests being served through the cash.

Table 4.6. Cash hit rate for the logistic regression model at each month, compared to the baseline and the Oracle.

Month	Baseline	LR	Oracle
2012/10	84.84%	84.88%	94.13%
2012/09	78.89%	80.19%	93.58%
2012/08	80.60%	81.82%	96.03%
2012/07	80.63%	81.94%	94.24%
2012/06	73.11%	66.30%	96.01%
2012/03	86.35%	86.55%	91.83%
2012/02	91.30%	91.06%	97.35%
2012/01	91.95%	90.81%	94.74%
2011/12	85.42%	82.69%	95.90%
2011/11	80.88%	81.47%	93.71%
2011/10	89.02%	90.34%	97.91%
2011/09	86.68%	88.37%	97.96%
2011/08	86.24%	87.52%	97.30%
2011/07	83.83%	85.45%	96.71%
2011/06	87.39%	88.19%	96.42%
2011/05	85.50%	85.98%	96.49%
2011/04	81.25%	81.39%	96.24%

4.5 Related Work

Previous research in web log analysis is as old as the web itself, and digital library log analysis can be traced back to before the web age with the TULIP project which ran from 1991 to 1995 [83]. Jamali et al. provided a survey about log analysis for digital libraries [84]. Jones et al. studied the transaction logs to a collection of computer science technical report comprising 46000 documents hosted on the New Zealand Digital Library [85]. Their work focused on queries and search patterns. The closest work to ours studied user behavior accessing Elsevier’s ScienceDirect in Taiwan [79]. Their work examined query patterns as well as downloading patterns. However, their analysis is limited to the Taiwanese population, unlike our analysis which considers a service with world wide user base. In addition, our analysis spans longer period and contains more access logs. More importantly, we model the access patters and fit it to a power law distribution. Furthermore, we notice the high traffic generated by the top 1% most downloaded papers and introduce a method to predict paper downloads.

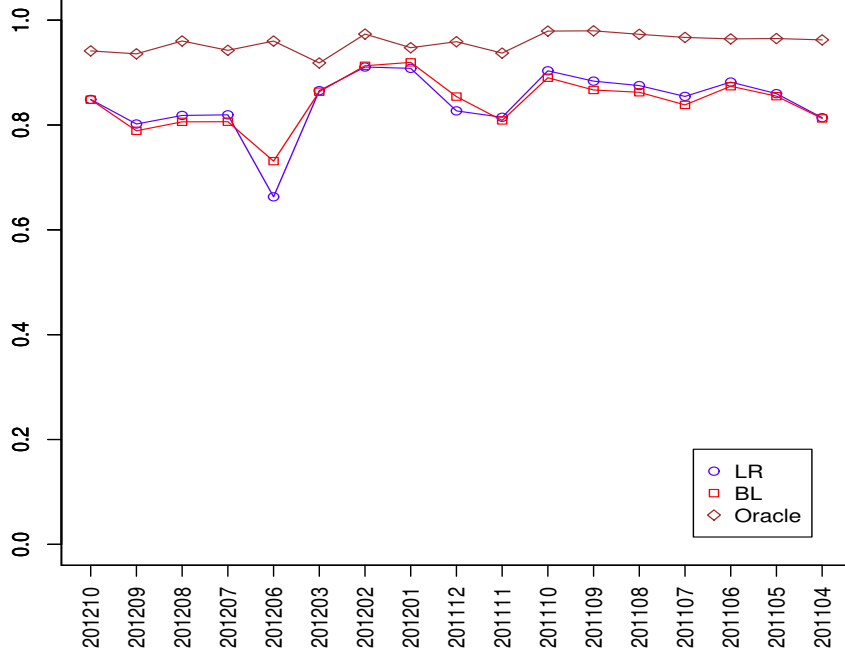


Figure 4.6. Cache hits of the top 1% most downloaded documents for each month. Each curve represents caching policy. LR refers to logistic regression, BL refers to base line. April and May of 2012 are omitted because very low traffic was recorded in them

Work load analysis based on the study of CiteSeer logs were performed earlier [86,87]. However it was conducted on a short time period of two weeks only. Our work aggregates data over three years, which allows us to observe behaviors not limited to short periods. In addition we introduce models to capture the observed behaviors. Download prediction was the topic of KDD Cup 2003 [88,89]. However it is different from the problem we tackle here, as the original KDD Cup challenge focused on predicting download counts in the first 60 days after the paper was published. Our problem is different as it focuses on monthly download prediction, rather than the short time after publication.

Search log analysis was studied before in the theme of Web search. Jansen et al. [76] studied the logs of less than 100,000 queries to Excite, while Taghavi and others [78] studied the query logs to multiple search engines by examining the requests going through a proxy server. Dogan et al. [74] studied the search

logs of Pubmed for a period of one month. Their work is only focused on the biomedical domain covered by Pubmed, whereas our analysis is on top of a computer and information sciences digital library. Furthermore, our analysis is on a much larger scale of three years, where in [74] the collected logs were for one month only. Earlier work has studied Pubmed queries on an even smaller scale of one day only [90]. Recently, book search behavior has been studied [91]. Although books can be part of a digital libraries, digital libraries contains different document types including papers that form a proceeding or a journal. Hence it is imperative to study the behavior at the paper level to get insight about each work, as users may be interested in specific sections of the book only.

Web caching of media resources have been investigated before for general web documents. Yang et al. combined association mining from web logs with standard caching algorithms to improve the hit rate [92]. Recker et al. used a window of 7 days to predict access logs on Georgia Tech network [93]. Smith and Wilbur used PubMed access logs to measure the popularity of articles and learn a model to predict popularity [94].

4.6 Conclusion and Future work

We have conducted a large scale analysis of access logs to the CiteSeerX digital library using logs from September 2009 to March 2013. Our analysis uses more than 3.3 billion log entries describing user queries, downloads, and other behavior. We have demonstrated that the number of downloads a paper receives can be modeled as a power law distribution. Also, we have showed that the top 1% most downloaded papers in a given month are likely to be highly downloaded in the following months. Furthermore, we observed a positive correlation, yet small, between the number of downloads and the number of citations for a given paper. A logistic regression model is introduced to predict the top 1% most downloaded articles in a given month. The model is shown to have improvement over simple baselines. Our analysis on the query logs shows a significant interest in author search, and advanced search more than typical full text search. In the future we plan to investigate better models to fit the download behavior associated with each paper and investigate new approaches to predict the number of downloads a paper receives.

Chapter 5 |

Academic Search

Academics have relied heavily on search engines to identify and locate research manuscripts that are related to their research areas. Many of the early information retrieval systems and technologies were developed while catering for librarians to help them sift through books and proceedings, followed by recent online academic search engines such as Google Scholar and Microsoft Academic Search. Despite of their popularity among academics and importance to academia, the usage, query behaviors, and retrieval models for academic search engines have not been well studied.

To this end, we study the distribution of queries that are received by an academic search engine, and analyze the click-through logs generated by the users. The click-through logs over a large period of time is found to positively correlate with an optimal ranking. Later we model the task of designing ranking function for academic queries as a learning to rank task, and show that it's effective to learn rankers. We also study the effect of multiple feature groups on the effectiveness of the learned ranking function, and realize that the article's full text is very important for the purpose of ranking. Furthermore, we delve deeper into academic search queries and classify them into navigational and informational queries. This work introduces a definition for navigational queries in academic search engines under which a query is considered navigational if the user is searching for a specific paper or document. We describe multiple facets of navigational academic queries, and introduce a machine learning approach with a set of features to identify such queries.

5.1 Introduction

Academic search engines have become the starting point for many researchers when they draft research manuscripts or work on proposals. Typically, there are two main retrieval systems that are used by academics. The first one is a citation database that is more of a traditional librarian search such as *Web of Science* and *Pubmed*, while the other is more similar to typical web search such as *Google Scholar*, and *Microsoft Academic Search*. Usage statistics tend to reflect user's preference for each type of systems. For example, 30% of Ph.D researchers relied on Google and Google Scholar as their main source for finding information in a survey conducted by the researchers of tomorrow project in 2012 [95]. On the other hand, usage statistics from the University of California, Santa Cruz indicate that Google Scholar was used as a secondary source of information rather than a primary one in 2010 [96].

Academics are different than regular web search users. First, they are either experts in the field they are searching about or they are being trained to become experts – in a graduate program. Secondly, it is typical for academic search to entail spanning beyond the first result page when researchers are performing literature review. Thirdly, recall can be of greater importance in academic search than of it in the case of typical web search. This is similar to legal retrieval where the fail to retrieve a relevant document may have significant repercussion. Furthermore, academics have good sense for judging the relevance of documents by examining certain information such as the number of citations a paper has received, the venue into which it was published, and author information. After all, algorithms such as *PageRank* [97] were inspired by the citation count model.

The need for retrieving relevant information in scientific domains have lead to many contributions in information retrieval [98]. For example, the idea behind indexing documents using keywords have originated from the field of librarianship [98,99]. Similarly, the intuition behind *PageRank* goes back to citation indexing, and using the number of citation as a proxy for measuring importance [97]. In addition, the *OHSUMED* dataset which became a standard benchmark for the quality of retrieval systems was built on top of *Medline* dataset [100].

In this chapter we focus on academic search systems that use keyword base search. We study user behavior of an academic search engine using query logs of

three years. By observing user query patterns, we were then motivated to study the user query intent by classifying academic search queries into navigational and informational queries. Search engine queries have typically been classified into three main categories [101, 102]: 1) Informational: the user is seeking some information available on the web; 2) Navigational: the user is trying to reach a particular website; and 3) Transactional: where the user is trying to perform some type of transaction. Query intent classification is an important part of any search engine for the effect of the query type has on the way it gets handled. Identifying navigational queries is essential for devising specific ranking functions that rank navigational queries only. Similarly, if a query is known to be navigational, the search engine may choose present the results page differently by either showing a single result or few results. In this work we introduce the concept of academic navigational query, and define multiple facets through which a query should be considered navigational. To the best of our knowledge, this is the first work that studies academic query classification.

Later we explore learning a ranking function that is suitable for academic search. In that regard, we observe a correlation between aggregate click through over time with the position at which the documents should be ranked, suggesting that aggregate click through rates can be effective in learning ranking functions for academic search. We hypothesize that this can be traced to the fact that academic search users are more knowledgeable than average web search users. Indeed, many academic search engine users are established researchers or PhD students with experience and training in the subject they are searching about.

Our experiments on learning a ranking function for academic search are conducted on top of manually judged dataset that contains 120 queries, with at least 20 documents tagged per query. The queries were sampled from real users of an academic search engine that indexes the full-text of the papers, along with title and abstract content. This is a clear distinction from other relevance datasets that were built on top of *Medline*, such as *OHSUMED*. In fact, including papers fulltext in the index and relevance judgment process reflects the real scenario when academics perform search tasks to retrieve documents. The lack of article fulltext in datasets such as *OHSUMED* makes the dataset not representative of real academic retrieval tasks. Therefore, as part of this work we will be later releasing the manually judged dataset for the public.

The contributions of the work include:

- We conduct the first study of academic search engines usage based on three years’ user query logs and show some significant uniqueness of academic search from general web search.
- We introduce the concept of academic navigational queries and the problem of academic query type classification. To explore the problem, we construct a new dataset using real queries and propose a set of features.
- We further study the feasibility of learning academic document ranking functions from user click-through data. We contribute a new dataset for IR community that contains real user queries and full text of academic documents, based on which, we investigate various ranking signals. Specially, we find full text features, which are usually ignored by academic search engines, play an important role.

5.2 Related Work

Although there is a rich literature related to academic search in both information retrieval and data mining community, few work studies the usage of academic search engines and seeks to understand academic search using real world user query data. We briefly review the related research on academic search in the following directions.

Existing academic search engines. Earlier academic search engines that build upon automatic citation indexing and metadata extraction include Citeseer [2, 103], followed by Arnetminer which focuses more on extraction, analysis and mining of academic social network and providing academic rankings [104, 105]. In the industry, two typical instances of academic search engines are Google Scholar and Microsoft Academic Search.

Academic document ranking. A key problem in academic search is ranking academic documents for user queries. Existing work studied general information retrieval models such as BM25, language models, HITS, PageRank, as well as academic specific metrics such as citations, venue impact and popularity factor, and author topics [106, 107]. However, they either lacked proper evaluation or used

a small private dataset. Our dataset is constructed by uniformly sampling from user queries of an academic search engine and we plan to make it publicly available. We also note that more recently Amolochitis etc. proposed a heuristic hierarchical scheme by combining heuristics including term-frequency, citation distribution and topics’ inter-relations [108]. However, their evaluation is conducted by making comparison to ACM digital library ranking results, thus it does not necessarily reflect the real world user query performance. We not only evaluate our ranking methods on human labeled query document dataset, but also show that a reasonable academic document ranker can be learned from user click through data.

Instead of ranking academic document alone, some work studied co-ranking or bi-type entity ranking for academic documents and authors [109], which estimates entity scores by considering both citation based link features and language model based content features in the bibliographic network. Besides, Zhou et al. studied similar work in a heterogeneous network based on coupling two random ranks on citation network and co-authorship network [107]. However, their evaluation is not based on real user queries, but synthetic queries that are selected from topical words generated by LDA models.

Query type identification. Most query type identification has focused on general Web search engines, with different methods applied in different settings. For example, Kang and Kim used the difference of distribution, mutual information, the usage rate as anchor texts, and the POS information for the classification [110]; Jansen et. al. proposed a rule based method based on a set of heuristics [111]; and Lee et. al. studied user-click behavior and anchor-link distribution [112]. However, to the best of our knowledge, no previous work studied identifying navigational queries in academic search.

Expert search. Similar to ranking authors given topics, a large amount of related work focused more on expert search [113]. Gollapalli etc. proposed a PageRank-like model that accommodates multiple sources of evidence for ranking experts in a particular topic [114]. Tang et al. presented a topic level expert search framework for heterogeneous networks by considering different entities in an academic network [115]. In this paper, our main focus is on academic document search, not author ranking or expert search, as we found most of the user queries are for document search.

Personalized academic search. Some research also investigated the person-

alized academic search by user modeling based on query log [116] or volunteer users [117]. It is worth noting that traditional user models such as collaborative filtering that do not consider document content features may fail since they play a more important role in ranking document than user similarity. Although we do not study personalized ranking in this work, we somehow show a consistent result: full text of academic documents matter a lot for some queries.

It is also worth noting that a research line on modeling the content of academic documents is author topic model, which associates each author with a multinomial distribution over topics and each topic is a multinomial distribution over words [118]. It has been applied to ranking authors by topics and providing interactive exploration of academic corpora [119]. Tang et al. also presented an author-venue-paper topic model and applied it on ArnetMiner [120]. However, the problem of the topic model based approach is that it might not be easily scaled to real world search engines due to the computation cost.

5.3 Academic Query Logs

Our work was motivated by examining the usage behavior of academic users of a major academic search engine that indexes publications in computer science and engineering, physics, and economics. The search engine provides multiple search types, most notably document search and author search. The search sessions received by the search engine between September of 2009 and March of 2013 are used to study multiple facets of user usage such as search types, query types, and document views. For deeper discussion into the query logs of the academic search engine, please refer to Chapter 4 of this dissertation.

5.4 Query Type Classification

Most of query intent identification has focused on general search engines with many approaches being applied [110–112]. However, there has not been any work that identifies navigational queries in academic search, to the best of our knowledge. In fact, we are not aware of any work that categorizes query intent in the academic search. Perhaps because traditional library search was the defacto standard in academic search until recently. The ease of use of current academic search engines

such as Google Scholar, combined with their similarity to traditional web search that most people have become familiar with creates an opportunity to study the query intent of the users.

In academic search it is possible to categorize queries into at least two types: *navigational* and *informational*. It is not straightforward how to define transactional queries, if they exist, in the academic setting. Providing a complete taxonomy of query intent in academic search is beyond the scope of this work. Rather, we focus on identifying navigational queries. We define a **navigational query** as a query for which the user is looking for a specific scholarly document, which can be a paper, book, thesis, etc. Correctly identifying navigational queries is important, because rankers are heavily influenced by citations which can lead to highly cited papers being ranked higher than the target paper if the target paper is new or not well cited. Furthermore, papers whose title contains general terms are more likely to be susceptible because there are large number of matches. There are multiple facets for navigational queries in academic search. For example, a user might look for a given document by:

- Document Object Identifier (DOI): *10.1038/nature14106*
- Full title query: *The Google file system*
- A combination of an author and title information: *jeff dean mapreduce*
- Author and year/venue information: *leskovec 2009 news cycle*
- Author names for a well known work: *Cormen Leiserson Rivest Stein*, or *hopcroft motwani ullman*
- A combination of author names along with some paper’s distinguishing terms: *dic brin motwani*

In the first scenario when users search by DOI, it is sufficient to check if the query matches a database of DOIs or not. However, other cases are not as trivial. For example, title queries are not easily detectable. First of all, extracting titles from papers is not always accurate. In addition, although a query can be checked against a list of titles, there are many short and ambiguous queries that might match at multiple title positions. On top of that, no search engines contain all academic documents [121], hence identifying a title navigational query that the search engine

Table 5.1. Navigational Query Features

Feature	Description
#_tokens	the number of tokens in the query
has_year	whether a term in the query matches a regex for identifying year
has_stop_word	whether the query has stop words
has_punctuation	whether the query has punctuation
#_authors	the number of tokens in the query identified as author name
author_ratio	the ratio of query terms identified as author names to the query length
is_title_match	whether the query matches a title in the search engine’s index

does not have a result for it may be used as signal to locate this missing document. However, other cases can be more subtle to identify. For example the following queries that were found in the logs of the academic search engine are not as obvious. In the query *leskovec 2009 news cycle* there exists an author name and a year along with subset of a title that would identify the work ¹. Similarly, *dic brin motwani* and *lift brin motwani ullman* both refer to the dynamic itemset counting paper by S. Brin and R. Motwani². Finding the correct matching of those queries might need more sophisticated approach than simple rules.

5.4.1 Approach

The problem is modeled as a binary classification problem. Given a query q , we would like to classify into one of the following classes {**Navigational**, **Informational**}. Each query q is represented as vector of the features described in Table 5.1. The features are crafted to capture the multiple facets that represent navigational queries. For example, **#_tokens** is chosen after noticing that many navigational queries have more terms than informational queries because they contain a title. On the other hand, **is_title_match** can be a good signal in general, but if the query term is general, a match does not necessarily make the query navigational.

¹Meme-tracking and the dynamics of the news cycle (KDD’09).

²Dynamic itemset counting and implication rules for market basket data (SIGMOD’97).

Other syntactic features such as `has_stop_word` , and `has_punctuation` are aimed at identifying title queries. The intuition is that users rarely use such terms in informational queries, and they are more likely to be part of title.

As shown in the examples, the mention of author names is one facet of navigational queries. However, it is not always the case that a mention of an author means a navigational query. For example, the following query that was found in the logs of the academic search engine is not considered navigational: `mccallum nigam`³ because these two authors have coauthored more than one paper together, and this query cannot be interpreted to refer to a single paper. Nevertheless, the presence of an author name is one of the indicators of navigational queries. Therefore, we create a feature to represent the number of query tokens that are identified to be an author name. Identifying whether a token refers to an author name or not was not as trivial as checking against a dictionary of all possible names. Initially we started by using the author list of DBLP⁴ as a names dictionary assuming that it would have low false positive rate since it is manually curated. However, the false positive identification was still high as tokens such as `network` matched as an author.

Therefore, we adapted a language model approach to identify author names. For every token t we estimate three probabilities: $P(t|author)$, $P(t|title)$, and $P(t|abstract)$ where author, title, and abstract refer to the token appearing in the author, title, or abstract section of the paper, respectively. We estimate each of the probabilities for every token over all the fields in the academic search engine’s index. A token t is considered an author iff:

$$P(t|author) > P(t|abstract) \wedge P(t|author) > P(t|title)$$

Gradient Boosted Trees (GBT) are used to train a classifier for identifying navigational queries. The number of stumps and the learning rate parameter are chosen using grid search over the range $[10, 400]$ and $[10^{-4}, 10^{-1}]$, respectively. SMOTE [122] oversampling is used to oversample the navigational queries because the dataset is imbalanced.

³Andrew McCallum and Kamal Nigam

⁴<http://www.dblp.org/search/index.php>

5.4.2 Dataset

To build the dataset, we first randomly sample 1000 queries from the user search logs and then keep only the queries in document search type, which results in 553 in total. However, notice that this small number of samples might not give reasonable coverage of all possible positive samples (navigational queries), we did multiple rounds of sampling and use the aforementioned heuristics to match possible positive candidates. The dataset was then augmented by those positive examples that might not have enough presence in the randomly sampled dataset, such as examples with author names. We also added comparable number of negative examples to counter for that effect. At the end, the dataset contained 579 queries. Each query was manually inspected by two human judges and tagged as either navigational or informational. When the judges had mismatching labels, they conferred and agreed on a mutual tag. In the manually tagged queries, 12.5% were found to be navigational. This percentage is smaller than that of general web queries as reported by Broder [101]. For queries tagged as informational, we found a large part of them are single word and phrase such as "802.3x", "iommu", and "nearest-neighbor based image classification", which actually refer to very specific concepts that might lead to only a small number of relevant results. That being said, it is possible that users are actually doing navigational search but using some simple informational queries, since they have a strong ability to quickly locate the best result from the returned list.

5.4.3 Experiments

The performance of the classifier is shown in Table 5.2. The numbers are for a 5 fold cross validation, with the training fold being randomly split into 90:10 with the 10% used to validate the grid search parameters. Oversampling using SMOTE was only conducted on the training fold, with the test fold remaining untouched. We compared the performance of the boosted tree classifier with that of an SVM with RBF kernel, and with that of a random forest. All parameters for both baseline classifiers are configured with grid search, similar to the GBT. The highest precision, and overall F score was attained with GBT as can be seen in table 5.2. The numbers in the table refer to the average precision, recall and F score obtained through the 5 fold cross validation, with the standard deviation

Table 5.2. Navigational query classification performance for multiple learning algorithms. Numbers between parenthesis refer to standard deviation in 5 fold cross validation.

Method	Precision	Recall	F1
GBT	0.68 (0.03)	0.68 (0.09)	0.677 (0.04)
SVM (RBF)	0.67 (0.05)	0.63 (0.12)	0.64 (0.07)
Random Forest	0.71 (0.06)	0.59 (0.14)	0.62 (0.09)

reported between parenthesis. The importance of each of the features is shown in Figure 5.1. The number of tokens within a query is the most important feature, which can be explained by title queries that tend to have higher number of tokens. Similarly, the title match feature which is closely related to the number of tokens in the query ranks second in terms of importance, followed by author ration feature.

It is worth noting that classifying navigational queries is notoriously a hard task in the web domain, and it would be at least as hard within the academic realm. For example, Jansen et al. [111] were only to obtain 74% for overall web query intent classification, which is not limited to navigational. Others were able to obtain 70% precision with high recall for informational queries [123]. Many studies for navigational query classification at web search engines have relied on click through rates [112, 124, 125]. While these methods were effective overall, they remain passive and depend on the presence of queries and clicks in the logs to be able to accurately classify them. This presents a challenge when new queries that have not been seen before arrive, or when users refer to an academic paper using a new combination of keywords.

5.5 Ranking

Id	Feature	Description
F1-4	tf_sum	Sum of all term frequencies. One feature per stream
F5-8	tf_max	Max term frequency within query. One feature per stream

F9-12	tf_min	Min term frequency within query. One feature per stream
F13-16	tf_avg	Average term frequency within query. One feature per stream
F17-20	tf_norm_sum	Sum of normalized term frequency. One feature per stream
F21-24	tf_norm_min	Min of normalized term frequency. One feature per stream
F25-28	tf_norm_max	Max of normalized term frequency. One feature per stream
F29-32	tf_norm_avg	Average of normalized term frequency. One feature per stream
F33-36	idf_sum	Sum of all term IDFs . One feature per steam
F37-40	idf_max	Max of all term IDFs . One feature per steam
F41-44	idf_min	Min of all term IDFs . One feature per steam
F45-48	idf_avg	Average of all term IDFs . One feature per steam
F49-52	tf_idf	Similarity using TFIDF . One feature per steam
F53-56	tf_idf_norm	Similarity using normalized TFIDF . One feature per steam
F57-60	bm25	Similarity using BM25 model. One feature per stream
F61-64	lm_jm	Similarity using language model smoothed using Jelinek-Mercer. One feature per stream
F65-68	lm_dr	Similarity using language model smoothed using Dirichlet smoothing. One feature per stream
F69-72	lm_as	Similarity using language model smoothed using absolute discount smoothing. One feature per stream
F73-76	num_covered	Number of covered query tokens in the document. One feature per stream

F77-80	ratio_covered	Ratio of covered query tokens in the document. One feature per stream
F81-84	stream_length	Length of stream. One feature per stream
F85	n_cites	number of citations an article received
F86	log_n_cites	log number of citations an article received

Table 5.3: The features used in representing every query document pair

Relevance ranking is one of the most important aspects of any retrieval system, and it is the case within an academic search engine as well. While academics have developed their own intuition about relative ranking of scholarly work through citations, impact factors, and other measures that allow them to use explicit aspects of the work to judge its quality and relevance, they would still benefit from a more general method for ranking. Our contribution for ranking in academic search is three fold. First, given that academic users are more knowledgeable than average web search users, we want to study if the wisdom of the crowd can be used to learn effective ranking functions. Previous work has shown that aggregate click-through logs over time can be used to learn ranking functions [126], despite presentational biases that make click orders unrepresentative of relevance within the session. In [126] it was found that click-through data over a period of time has 0.20 Kendall tau-b correlation with labels of human judges, when considering clicked documents only. When unclicked documents were added, the correlation became 0.35. Later, the click-through data was fed as training examples to a learning to rank algorithm that achieved competitive NDCG. We want to study if this would be the case for academic search. Secondly, we want to study the effectiveness of learning a ranking function for academic search, and what class of features would have a significant contribution. Thirdly, we are creating a dataset of human judged queries and papers that can be used not only for academic search, but also as benchmark for learning to rank. We believe that our dataset is more representative of academic search than other benchmarks such as *OHSMUND* because it contains article’s full text as well.

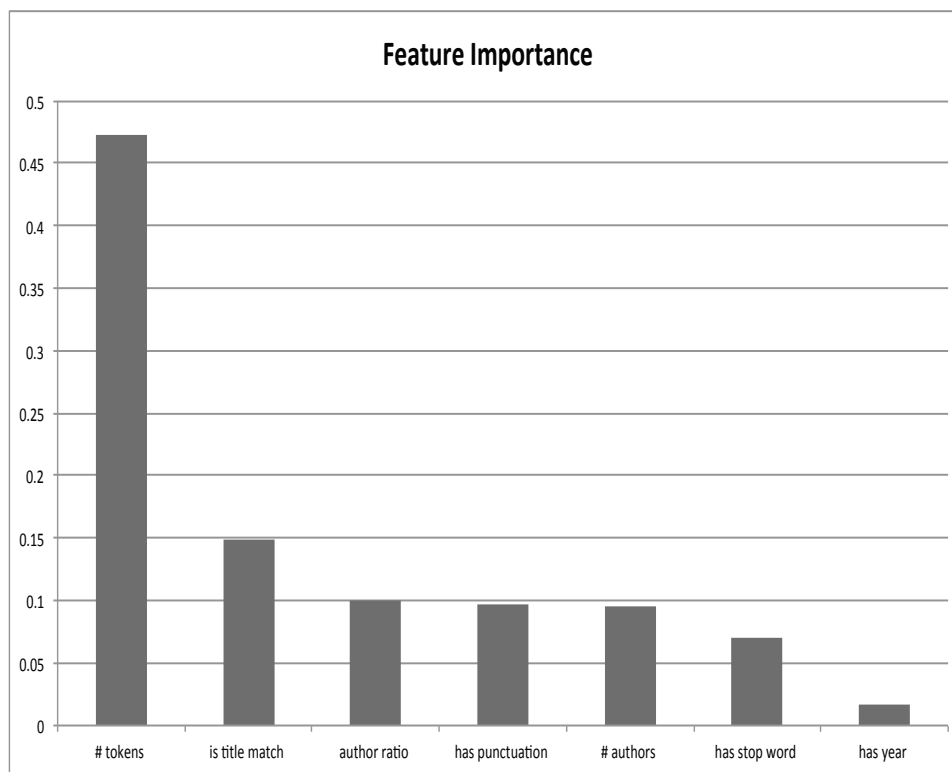


Figure 5.1. Feature importance of navigational query classification. Number of tokens does contribute the most but all other features give reasonable contributions.

5.5.1 Dataset

Our dataset contains 120 queries that were randomly sampled from the access logs of the academic search engine where for each query there exist at least one clicked result. Again, we only consider queries from document search type since it is the majority search type in our system. All the list of documents that were ever clicked on as a result of a query were manually judged by two human judges using 5 level grades: Perfect, Good, Fair, Not Good, Bad.

- Perfect: This paper is what exactly I am looking for and I will read it immediately.
- Good: This paper is relevant to the query and I would like to read it.
- Fair: The paper seems relevant to some extent and I need to do further check to see if I need it.

- Not Good: The paper might be a little relevant but it is not what I need.
- Bad: The paper is not relevant at all and I definitely do not need it.

For each query and document, the judges were presented by the query text, document title, abstract, authors, and number of citations. The number of documents that were tagged for each query varied between 3 up to 100.

The kappa inter annotator agreement between the human judges was 0.39. This is a reasonable measure given that binary scale annotations typically have a kappa statistic in the range (0.67, 0.8) [127]. However, the value of the kappa statistic was 0.79 when allowing the ratings of the two judges to vary by 1 point only. Overall, there are 2344 query document pairs that were manually judged in our dataset.

5.5.2 Labels and Clickthrough Rates

In this section we study whether the aggregate clickthrough counts of documents for a given query agree with the human judges labels. Kendal tau test statistic is used to estimate the agreement between the two ranked lists, which measures the agreement between two ranked lists by counting the number of concordant pairs, along with the number of discordant pairs. For every query q and document d_i , we define $label(q, d_i)$ as the label assigned to document d_i when querying for query q by the judges. Similarly, we define $click(q, d_i)$ as the number of times document d_i was clicked as result of searching for q . Hence, given the documents d_i and d_j that are both annotated for query q we define:

- d_i and d_j are concordant if (1) $click(q, d_i) > click(q, d_j)$ and $label(q, d_i) > label(q, d_j)$, or (2) $click(q, d_i) < click(q, d_j)$ and $label(q, d_i) < label(q, d_j)$
- d_i and d_j are discordant if (1) $click(q, d_i) > click(q, d_j)$ and $label(q, d_i) < label(q, d_j)$, or (2) $click(q, d_i) < click(q, d_j)$ and $label(q, d_i) > label(q, d_j)$
- d_i and d_j are tied otherwise

The Kendall Tau rank correlation statistic is a value between -1 and 1, with 1 meaning complete agreement, and -1 meaning complete negative association. We found a positive correlation of 0.34 when considering queries for which KT correlation is defined. This is a significantly larger positive correlation than that

was found between number of clicks and relevance in general web search. In fact, it was found that only by treating unclicked results as irrelevant document that the correlation in web search would be 0.35 [126]. This further supports our assumption that academic search users are in fact more knowledgeable than average web search users. Furthermore, since it was possible to learn effective ranking results based simply on click counts with 0.20 correlation for general web search [126], it would be more effective to learn a ranking function in academic search because there is a stronger correlation between the absolute click count and the relevance.

5.5.3 Experiments

In this section we study the effectiveness of learning to rank approaches on academic queries. We devise the features presented in Table 5.3 to represent each query document pair. Each document consists of four streams: title, abstract, authors, and fulltext. The title stream refers to the automatically identified title of the paper; similarly for the abstract and authors. All features are generated per stream, except for citation features (F85 and F86) that are per document only. For example, features 1-4 consist of the sum of the term frequencies within each of streams. The first group of features (F1 - 48) contains statistics about term frequencies and inverse document frequencies for the terms appearing in the query. Features 49-56 represent the similarity between the query and each of the streams using TFIDF and weighted TFIDF. For BM25 in features 57-60 we set $k_1 = 2.5$ and $b = 0.8$. We use language model similarity using three smoothing methods in features 61-72. Jelinek-Mercer smoothing is used in features 61-64 with $\lambda = 0.7$. This value was found to be optimal for longer queries [128], which is the case in academic search with average query length being longer than web search. Dirichlet smoothing is used in features 65-68 with μ set to the average stream length. In features 69-72, the absolute discount smoothing is used with $\delta = 0.7$ per [128]. Finally, features 77-84 capture the number of covered query tokens within each streams.

We have considered multiple learning to rank algorithms, but obtained the best performance when using LambdaMART⁵ [129]. [130] LambdaMART is a boosted tree version of LambdaRank [131] using MART [132]. The parameters of LambdaMART ranking function are set as follows: number of trees = 100, number

⁵We used RankLib implementation <http://sourceforge.net/p/lemur/wiki/RankLib/>

of leaves per tree = 20, learning rate = 0.2, and minimum number of instances in a leaf is 10. The experiments are conducted in 5 fold cross validation where the algorithm is trained to maximize Normalized Discounted Cumulative Gain at 10 (NDCG@10).

$$NDCG@K = \frac{1}{DCG@K_{OPT}} \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i + 1)}$$

where $DCG@K_{OPT}$ corresponds to the discounted gain for an optimal ranking. For queries that had less than 20 judged documents, we randomly sampled documents that match the query from the index, but were not judged before and gave them a relevant judgment of 0. Unless we do that, then accurately ranking a list of 3 documents would be equal to ranking a list of 50 documents, which we wanted to avoid because it would artificially boost overall average NDCG. We have also removed 8 queries that we found to be very general and ambiguous, such as *cisco*, *certificate*, and *bibtex*.

The results of the 5 fold cross validation of the LambdaMART ranker along with multiple baselines are reported in Table 5.4. In all the results, LambdaMART is trained to optimized NDCG@10, regardless of the test measure. The best performing model that utilizes all the features achieves NDCG@1 of 0.64, well better than any other baseline. We consider multiple baselines that included subset of the features with the weights being also learned using LambdaMART. The first baseline is a BM25 model, with 4 features (F 57-60 in Table 5.3). Another baseline adds citation features (F85 - 86) to the base BM25 model. It is obvious from the results that a model with all the features outperforms a BM25 learned model. Similarly we used a ranker that utilizes language model features only (F 61-72) as a baseline, which is also shown in Table 5.4 to have lower NDCG than the model with all features. In addition, we compare against a TFIDF ranking function baseline that includes features 49-56 along with citation features. The model with all the features consistently outperformed all the baselines across multiple values of k when testing for $NDCG@k$.

We also study the contribution of certain feature classes to the best obtained ranker. To test the importance of a given feature group, LambdaMART is used to learn a ranker using all the features except this given feature group. Through this process we realize two observations. First, citations are very important for

learning a ranking function in academic search, with NDCG@1 dropping by 5%, and NDCG@3 dropping 3%. This is expected because academics have always used citations to measure impact and quality. In addition to citations, the full text of the papers is very important to learn a ranking function. In fact NDCG@1 would drop by 4% when full text features are dropped. Similarly, NDCG@3 would drop by 3%. This finding is of importance to both users of academic search engines and for developers of the search engines. Based on this, it is imperative to include the papers full text into the index to be able to learn a better ranking function. Similarly, users of academic search engines should prefer to use engines which indexes the full text of the academic papers for it would be able to present more relevant results to the users.

These results would suggest that the more the academic papers are open and available to academic search engines, the better they will be able to cater to user needs. With policies and subscription governing the access to publisher's papers, it is challenging to devise effective ranking functions. Furthermore, our findings suggest that open access would have positive impact on the discoverability of research articles, thus saving researchers time locating articles.

Table 5.4. The performance obtained at various levels by multiple feature set combinations and baselines by applying LambdaMART algorithm. Training is optimized for NDCG@10 in all cases. All - [feature] denotes using all features except for the group listed between brackets. Plus sign denotes using only the listed features.

Features	NDCG@1	NDCG@3	NDCG@5	NDCG@10	NDCG@20
All	0.6484	0.6744	0.6952	0.742	0.8247
All - fulltext	0.6063	0.6478	0.6843	0.7351	0.8157
All - citations	0.5916	0.6436	0.6608	0.7116	0.8059
BM25	0.5987	0.6275	0.6368	0.699	0.7975
BM25 + citations	0.5677	0.6294	0.6546	0.7142	0.7996
Language Model	0.5908	0.6237	0.6402	0.69	0.7938
Language Model + citations	0.5677	0.6436	0.6666	0.7265	0.8056
TFIDF	0.5399	0.6053	0.6263	0.6845	0.7828
TFIDF + citations	0.5359	0.6192	0.6504	0.713	0.7954

5.6 Conclusion and Future Work

We studied academic search based on user query logs and investigated two important IR tasks: academic query type classification and document ranking. We started by showing some interesting statistics and findings mined from three years' real world user query history. We found that 1) document and author search are the two main types for academic search while document search is the majority; 2) distribution of query length follows a long tail with an average of 3.85, which is longer than general web queries; 3) the majority (92+%) of users only looked at the first result page but user were more likely to look at 5 results pages than at 2, 3, and 4; and 4) the user clicks would be more likely follow the cascade model than the multiple browsing model. We then introduced the concept of academic navigational query and studied the problem of academic query type classification on a new dataset with human judgments. To the best of our knowledge, this was the first attempt at classifying academic search queries. We then proposed a set of features to learn the classifier. The results showed the effectiveness of the proposed features and demonstrated the challenge of the problem. In addition, we studied the academic document ranking using real world user queries. Specifically, we presented a ranking model for academic document search that leverages multiple ranking signals and contributed a new dataset for ranking with 5 level judgment on real user queries. In particular, we found that adding full text features would be helpful for the academic document ranking, which indicates the importance of full text indexing for academic search engines. Although our results also reconfirm the importance of citation based features for ranking academic papers, a thorough experimental evaluation shows that our learned model outperforms the strong baselines such as BM25 or language model along with citation features. Our findings suggest that academic search engines that are able to index the full text of the articles might be able to rank relevant results higher.

We have several promising future directions. One of our ongoing work is on implementing our new academic ranking methods in to the academic search engine by taking advantage of both navigational query classification and the learned academic document rank functions. More specifically, we could add the navigational query classification result as a new feature for the learning to rank model or we could train separate rankers for different types of queries. Besides, we plan to

explore more possibility on using much larger scale user click-through data to train rankers periodically in an automatic way. However, there would be challenges on data selection and cleaning. Another possible future work is to explore the personalized academic search using the query logs for different academic entities such as documents and authors.

Chapter 6 |

Conclusion and Future Work

This dissertation has studied the accessibility of scholarly data in general, and scholarly documents in specific. It first started by obtaining an estimate of the total number of scholarly documents on the web. Capture-recapture methods are used by considering two academic search engines to be a random capture of the scholarly documents on the web. Using both Google Scholar and Microsoft Academic Search it is estimated that at least 114 million scholarly documents are available on the web. Furthermore, estimates are obtained for each scientific discipline. Later, the percentage of openly accessible scholarly documents are estimated using the coverage of Google Scholar where it is found that this percentage varies significantly across scientific disciplines. Nevertheless, we estimate that at least 24% of the scholarly documents can be found freely on the web.

After that we focus on extracting chemical formulae and names from chemistry documents as an example of information extraction that provides better accessibility to sub objects of scholarly documents. Two methods are introduced. The first approach is a probabilistic framework for ensemble extraction where the output of multiple independent extractors are combined together probabilistically. In the second approach a conditional random fields extractor is used with a combination of novel features that include phonetic encoding through the Soundex algorithm, in addition to utilizing word embeddings. Furthermore, we provide a study on the effect of tokenizing on the accuracy of the overall extraction process where we show that it can be used as an upper bound of the the attainable recall in the absence of any post processing.

Later we study the user interactions with the CiteSeerX digital library and search engine by mining the access logs of more than 3 years. Through the log

analysis we are able to gain a better understanding of user behavior which can be used in designing next generation repositories and search engine. We also model the session length and the number of downloads a given paper receives over a time as a power law distribution. In addition, we notice that the highly accessed papers present a repetitive pattern where they remain highly accessible in the following months. We capitalize on that by showing that the highly accessed documents in a given month can be predicted using a logistic regression model that relies on historical data as features.

In the final chapter of this thesis we focused on studying academic search to aid the discovery of scholarly documents. First we categorized user queries based on the user's intent into two classes: navigational and informational queries, where we defined navigational queries. Later we showed how users may have a navigational intention using variety of forms. A gradient boosting tree method is used to identify navigational queries from the majority of informational ones using multiple features. Furthermore, we show that user clicks in academic search correlate positively with the optimal ranking that documents should be ranked according to. Later, we focus on devising a ranking function for academic search using a newly created dataset of human judgements. The impact of certain features on the performance of the ranker is studied as well.

6.1 Future Work

Building on the work described within this dissertation can take multiple avenues. First, our work on size estimates was limited to English documents only. It would be interesting to get an estimate of the total number of scholarly documents in all languages. In addition, a longitudinal study would provide an estimate to the annual growth of scholarly documents on the web. Developing another sampling method would be a welcomed addition to the literature, and a contribution that may have an effect beyond scholarly domain itself, and rather into other problems that rely on sampling to get size estimates.

For log mining, there is tremendous opportunity to unveil even more interesting observations. In our work we have relied on historical access logs that were collected using default configuration of web servers. However, by tracking users through browser cookies it would be possible to track user preference over time and link

sessions by the user. Furthermore, new scholarly documents may be recommended to the user based on her personal click through history. Aside from the users, individual paper popularity can be traced over time. Previous work has shown that download patterns can be used as an early sign of citations, but the distribution of downloads over time for a given paper can be used to learn more about the life cycle of a given scholarly document. There is also room for improvement when it comes to predicting highly accessed papers. Perhaps, a time series model might be explored here.

Academic search may also benefit from fine grained user tracking which may reveal relation between search sessions. Better indicators and methods for identifying query intent in academic search should be explored. To the best of our knowledge, our work is the first to make the distinction between query goals in academic search, and it should open the door for further explorations. Similarly, novel signals may be introduced to learn a more effective ranking functions.

Bibliography

- [1] NOORDEN, R. V. (2013) “Open access: The true cost of science publishing,” *Nature*, **495**(7442), pp. 426–429.
- [2] 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*, ACM, pp. 89–98.
- [3] Web of Science fact page: <http://wokinfo.com/realfacts/qualityandquantity/>.
- [4] Based on the statistics reported at the homepage of Microsoft Academic Search as of January 10, 2013: <http://academic.research.microsoft.com>.
- [5] BAR-ILAN, J. (2008) “Which h-index? A comparison of WoS, Scopus and Google Scholar,” *Scientometrics*, **74**(2), pp. 257–271.
- [6] ——— (2010) “Citations to the “Introduction to informetric” indexed by WOS, Scopus and Google Scholar,” *Scientometrics*, **82**(3), pp. 495–506.
- [7] BJÖRK, B.-C., A. ROOS, and M. LAURI (2009) “Scientific journal publishing—yearly volume and open access availability,” *Information Research*, **14**(1), p. 391.
- [8] LAWRENCE, S. and C. GILES (1998) “Searching the world wide web,” *Science*, **280**(5360), pp. 98–100.
- [9] ——— (1999) “Accessibility of information on the web,” *Nature*, **400**(6740), pp. 107–9.
- [10] BHARAT, K. and A. BRODER (1998) “A technique for measuring the relative size and overlap of public web search engines,” *Computer Networks and ISDN Systems*, **30**(1), pp. 379–388.
- [11] DOBRA, A. and S. E. FIENBERG (2004) “How large is the world wide web,” *Web Dynamics*, pp. 23–44.

- [12] BRODER, A. Z., S. C. GLASSMAN, M. S. MANASSE, and G. ZWEIG (1997) “Syntactic clustering of the web,” *Computer Networks and ISDN Systems*, **29**(8), pp. 1157–1166.
- [13] NORRIS, M., C. OPPENHEIM, and F. ROWLAND (2008) “The citation advantage of open-access articles,” *Journal of the American Society for Information Science and Technology*, **59**(12), pp. 1963–1972.
- [14] HOGG, R. and E. TANIS (2010) *Probability and Statistical Inference*, Pearson/Prentice Hall.
- [15] GARGOURI, Y., C. HAJJEM, V. LARIVIÈRE, Y. GINGRAS, L. CARR, T. BRODY, and S. HARNAD (2010) “Self-selected or mandated, open access increases citation impact for higher quality research,” *PloS ONE*, **5**(10), p. e13636.
- [16] HAJJEM, C., S. HARNAD, and Y. GINGRAS (2005) “Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact,” *IEEE Data Engineering Bulletin*, **28**(4), pp. 39–47.
- [17] BJÖRK, B.-C., P. WELLING, M. LAAKSO, P. MAJLENDER, T. HEDLUND, and G. GUÐNASON (2010) “Open access to the scientific journal literature: situation 2009,” *PloS one*, **5**(6), p. e11273.
- [18] PAGE, L., S. BRIN, R. MOTWANI, and T. WINOGRAD (1999) *The PageRank Citation Ranking: Bringing Order to the Web., Technical Report 1999-66*, Stanford InfoLab.
- [19] LINCOLN, F. C. (1930) “Calculating Waterfowl Abundance on the Basis of Banding Returns,” *US Department of Agriculture Circular*, **118**.
- [20] PETERSEN, C. (1896) “The yearly immigration of young plaice into the Limfjord from the German Sea,” *Report of the Danish Biological Station*, **6**, pp. 1–48.
- [21] SHELDON, R. ET AL. (2009) *A First Course In Probability, 8/E*, Pearson Education.
- [22] BAILLARGEON, S. and L.-P. RIVEST (2007) “Rcapture: loglinear models for capture-recapture in R,” *Journal of Statistical Software*, **19**(5), pp. 1–31.
- [23] CORMACK, R. M. (1989) “Log-linear models for capture-recapture,” *Biometrics*, pp. 395–413.
- [24] CORMACK, R. and P. JUPP (1991) “Inference for Poisson and multinomial models for capture-recapture experiments,” *Biometrika*, **78**(4), pp. 911–916.

- [25] CORMACK, R. (1992) “Interval estimation for mark-recapture studies of closed populations,” *Biometrics*, pp. 567–576.
- [26] AKAIKE, H. (1974) “A new look at the statistical model identification,” *Automatic Control, IEEE Transactions on*, **19**(6), pp. 716–723.
- [27] CRAVEN, M., A. MCCALLUM, D. PIPASQUO, T. MITCHELL, and D. FREITAG (1998) *Learning to extract symbolic knowledge from the World Wide Web*, *Tech. rep.*, DTIC Document.
- [28] COHEN, W. W. and Y. SINGER (1999) “A simple, fast, and effective rule learner,” in *Proceedings of the National Conference on Artificial Intelligence*, John Wiley & Sons Ltd, pp. 335–342.
- [29] MCCALLUM, A., D. FREITAG, and F. C. PEREIRA (2000) “Maximum Entropy Markov Models for Information Extraction and Segmentation.” in *ICML*, pp. 591–598.
- [30] LAFFERTY, J., A. MCCALLUM, and F. C. PEREIRA (2001) “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” .
- [31] SUTTON, C. and A. MCCALLUM (2006) *An introduction to conditional random fields for relational learning*, vol. 2, Introduction to statistical relational learning. MIT Press.
- [32] CORBETT, P. and P. MURRAY-RUST (2006) “High-throughput identification of chemistry in life science texts,” in *Computational Life Sciences II*, Springer, pp. 107–118.
- [33] SUN, B., Q. TAN, P. MITRA, and C. L. GILES (2007) “Extraction and search of chemical formulae in text documents on the web,” in *Proceedings of the 16th international conference on World Wide Web*, ACM, pp. 251–260.
- [34] SUN, B., P. MITRA, and C. L. GILES (2008) “Mining, indexing, and searching for textual chemical molecule information on the web,” in *Proceedings of the 17th international conference on World Wide Web*, ACM, pp. 735–744.
- [35] SUN, B., P. MITRA, C. LEE GILES, and K. T. MUELLER (2011) “Identifying, Indexing, and Ranking Chemical Formulae and Chemical Names in Digital Documents,” *ACM Transactions on Information Systems (TOIS)*, **29**(2), p. 12.
- [36] JESSOP, D. M., S. E. ADAMS, E. L. WILLIGHAGEN, L. HAWIZY, and P. MURRAY-RUST (2011) “OSCAR4: a flexible architecture for chemical text-mining,” *Journal of cheminformatics*, **3**(1), pp. 1–12.

- [37] ROCKTÄSCHEL, T., M. WEIDLICH, and U. LESER (2012) “ChemSpot: a hybrid system for chemical named entity recognition,” *Bioinformatics*, **28**(12), pp. 1633–1640.
- [38] KRALLINGER, M., F. LEITNER, O. RABAL, M. VAZQUEZ, J. OYARZABAL, and A. VALENCIA (2013) “Overview of the chemical compound and drug name recognition (CHEMDNER) task,” in *BioCreative Challenge Evaluation Workshop vol. 2*, p. 2.
- [39] KHABSA, M. and C. L. GILES (2013) “An Ensemble Information Extraction Approach to the BioCreative CHEMDNER Task,” in *BioCreative Challenge Evaluation Workshop vol. 2*, p. 105.
- [40] “OSCAR 4,” Last accessed 9/19/13.
URL <https://bitbucket.org/wmm/oscar4/wiki/Home>
- [41] “ChemSpot,” Last accessed 9/17/13.
URL <https://www.informatik.hu-berlin.de/forschung/gebiete/wbi/resources/chemspot/chemspot>
- [42] SOHN, S., D. C. COMEAU, W. KIM, and W. J. WILBUR (2008) “Abbreviation definition identification based on automatic precision estimates,” *BMC bioinformatics*, **9**(1), p. 402.
- [43] “Reflect,” Last accessed 9/19/13.
URL <http://reflect.ws/>
- [44] “Whatizit,” Last accessed 9/19/13.
URL <http://www.ebi.ac.uk/webservices/whatizit/info.jsf>
- [45] “MiniChem,” Last accessed 9/18/13.
URL http://vega.soi.city.ac.uk/~abdy181/software/GATE/MiniChem_Tagger/creole.zip
- [46] “Lucene,” Last accessed 3/25/14.
URL <http://lucene.apache.org/>
- [47] WOLPERT, D. H. (1992) “Stacked generalization,” *Neural networks*, **5**(2), pp. 241–259.
- [48] FLORIAN, R. (2002) “Named entity recognition as a house of cards: Classifier stacking,” in *proceedings of the 6th conference on Natural language learning-Volume 20*, Association for Computational Linguistics, pp. 1–4.
- [49] LEAMAN, R., C.-H. WEI, and Z. LU (2013) “NCBI at the BioCreative IV CHEMDNER Task: Recognizing chemical names in PubMed articles with tmChem,” in *BioCreative Challenge Evaluation Workshop vol. 2*, p. 34.

- [50] YOSHIOKA, M. and T. M. DIEB (2013) “Ensemble Approach to Extract Chemical Named Entity by Using Results of Multiple CNER Systems with Different Characteristic,” in *BioCreative Challenge Evaluation Workshop vol. 2*, p. 162.
- [51] HUBER, T., T. ROCKTÄSCHEL, M. WEIDLICH, P. THOMAS, and U. LESER (2013) “Extended Feature Set for Chemical Named Entity Recognition and Indexing,” in *BioCreative Challenge Evaluation Workshop vol. 2*, p. 88.
- [52] FLORIAN, R., A. ITTYCHERIAH, H. JING, and T. ZHANG (2003) “Named entity recognition through classifier combination,” in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, pp. 168–171.
- [53] MCCALLUM, A. K. (2002) “MALLET: A Machine Learning for Language Toolkit,” [Http://mallet.cs.umass.edu](http://mallet.cs.umass.edu).
- [54] “Apache OpenNLP,” Last accessed 3/25/14.
URL <http://opennlp.apache.org/>
- [55] BROWN, P. F., P. V. DESOUZA, R. L. MERCER, V. J. D. PIETRA, and J. C. LAI (1992) “Class-based n-gram models of natural language,” *Computational linguistics*, **18**(4), pp. 467–479.
- [56] MIKOLOV, T., K. CHEN, G. CORRADO, and J. DEAN (2013) “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*.
- [57] “Soundex,” Last accessed 3/25/14.
URL <http://www.archives.gov/research/census/soundex.html>
- [58] LEAMAN, R., G. GONZALEZ, ET AL. (2008) “BANNER: an executable survey of advances in biomedical named entity recognition.” in *Pacific Symposium on Biocomputing*, vol. 13, pp. 652–663.
- [59] HETTNE, K. M., R. H. STIERUM, M. J. SCHUEMIE, P. J. HENDRIKSEN, B. J. SCHIJVENAARS, E. M. VAN MULLIGEN, J. KLEINJANS, and J. A. KORS (2009) “A dictionary to identify small molecules and drugs in free text,” *Bioinformatics*, **25**(22), pp. 2983–2991.
- [60] MORSE, D. H. and W. A. CLINTWORTH (2000) “Comparing patterns of print and electronic journal use in an academic health science library,” *Issues in Science and Technology Librarianship*, **28**(26.04), p. 2008.
- [61] DE GROOTE, S. L. and J. L. DORSCH (2001) “Online journals: impact on print journal usage,” *Bulletin of the Medical Library Association*, **89**(4), p. 372.

- [62] ——— (2003) “Measuring use patterns of online journals and databases,” *Journal of the Medical Library Association*, **91**(2), p. 231.
- [63] JOACHIMS, T. (2002) “Optimizing search engines using clickthrough data,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 133–142.
- [64] XIANG, R., J. NEVILLE, and M. ROGATI (2010) “Modeling relationship strength in online social networks,” in *Proceedings of the 19th international conference on World wide web*, ACM, pp. 981–990.
- [65] OLSTON, C., B. REED, U. SRIVASTAVA, R. KUMAR, and A. TOMKINS (2008) “Pig latin: a not-so-foreign language for data processing,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM, pp. 1099–1110.
- [66] “Apache Pig,” <http://pig.apache.org/>.
- [67] THUSOO, A., J. S. SARMA, N. JAIN, Z. SHAO, P. CHAKKA, S. ANTHONY, H. LIU, P. WYCKOFF, and R. MURTHY (2009) “Hive: a warehousing solution over a map-reduce framework,” *Proceedings of the VLDB Endowment*, **2**(2), pp. 1626–1629.
- [68] “Apache Hive,” <http://hive.apache.org/>.
- [69] RADLINSKI, F. and T. JOACHIMS (2005) “Query chains: learning to rank from implicit feedback,” in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, ACM, pp. 239–248.
- [70] EICKHOFF, C., J. TEEVAN, R. WHITE, and S. DUMAIS (2014) “Lessons from the Journey: A Query Log Analysis of Within-Session Learning,” in *To appear in WSDM 2014*.
- [71] TANASA, D. and B. TROUSSE (2004) “Advanced data preprocessing for intersites web usage mining,” *Intelligent Systems, IEEE*, **19**(2), pp. 59–65.
- [72] CLAUSET, A., C. R. SHALIZI, and M. E. NEWMAN (2009) “Power-law distributions in empirical data,” *SIAM review*, **51**(4), pp. 661–703.
- [73] KUNEGIS, J. and J. PREUSSE (2012) “Fairness on the web: Alternatives to the power law,” in *Proceedings of the 3rd Annual ACM Web Science Conference*, ACM, pp. 175–184.
- [74] DOGAN, R. I., G. C. MURRAY, A. NÉVÉOL, and Z. LU (2009) “Understanding PubMed® user search behavior through log analysis,” *Database*, **2009**, p. bap018.

- [75] SILVERSTEIN, C., M. HENZINGER, H. MARAIS, and M. MORICZ (1998) *Analysis of a very large altavista query log*, *Tech. rep.*, Technical Report 1998-014, Systems Research Center, Compaq Computer Corporation.
- [76] JANSEN, B. J., A. SPINK, J. BATEMAN, and T. SARACEVIC (1998) “Real life information retrieval: A study of user queries on the web,” in *ACM SIGIR Forum*, vol. 32, ACM, pp. 5–17.
- [77] ZHANG, Y., B. J. JANSEN, and A. SPINK (2009) “Time series analysis of a Web search engine transaction log,” *Information Processing & Management*, **45**(2), pp. 230–245.
- [78] TAGHAVI, M., A. PATEL, N. SCHMIDT, C. WILLS, and Y. TEW (2012) “An analysis of web proxy logs with query distribution pattern approach for search engines,” *Computer Standards & Interfaces*, **34**(1), pp. 162–170.
- [79] KE, H.-R., R. KWAKKELAAR, Y.-M. TAI, and L.-C. CHEN (2002) “Exploring behavior of e-journal users in science and technology: transaction log analysis of Elsevier’s ScienceDirect OnSite in Taiwan,” *Library & Information Science Research*, **24**(3), pp. 265–291.
- [80] CRASWELL, N., O. ZOETER, M. TAYLOR, and B. RAMSEY (2008) “An experimental comparison of click position-bias models,” in *Proceedings of the 2008 International Conference on Web Search and Data Mining*, ACM, pp. 87–94.
- [81] DUPRET, G. E. and B. PIWOWARSKI (2008) “A user browsing model to predict search engine click data from past observations,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 331–338.
- [82] GILBERT, J. C. and C. LEMARÉCHAL (2006) *Numerical optimization: theoretical and practical aspects*, Springer.
- [83] BORGHUIS, M. ET AL. (1996) *TULIP: final report*, vol. 18, Elsevier Science New York.
- [84] JAMALI, H. R., D. NICHOLAS, and P. HUNTINGTON (2005) “The use and users of scholarly e-journals: a review of log analysis studies,” in *Aslib Proceedings*, vol. 57, Emerald Group Publishing Limited, pp. 554–571.
- [85] JONES, S., S. J. CUNNINGHAM, R. MCNAB, and S. BODDIE (2000) “A transaction log analysis of a digital library,” *International Journal on Digital Libraries*, **3**(2), pp. 152–169.

- [86] LI, H., W.-C. LEE, A. SIVASUBRAMANIAM, and C. L. GILES (2007) “A hybrid cache and prefetch mechanism for scientific literature search engines,” in *Web Engineering*, Springer, pp. 121–136.
- [87] ——— (2008) “Workload analysis for scientific literature digital libraries,” *International Journal on Digital Libraries*, **9**(2), pp. 139–149.
- [88] BRANK, J. and J. LESKOVEC (2003) “The download estimation task on KDD Cup 2003,” *ACM SIGKDD Explorations Newsletter*, **5**(2), pp. 160–162.
- [89] GEHRKE, J., P. GINSPIRG, and J. KLEINBERG (2003) “Overview of the 2003 KDD Cup,” *ACM SIGKDD Explorations Newsletter*, **5**(2), pp. 149–151.
- [90] HERSKOVIC, J. R., L. Y. TANAKA, W. HERSH, and E. V. BERNSTAM (2007) “A day in the life of PubMed: analysis of a typical day’s query log,” *Journal of the American Medical Informatics Association*, **14**(2), pp. 212–220.
- [91] KIM, J. Y., H. FEILD, and M. CARTRIGHT (2012) “Understanding book search behavior on the web,” in *Proceedings of the 21st ACM international conference on Information and knowledge management*, ACM, pp. 744–753.
- [92] YANG, Q., H. H. ZHANG, and T. LI (2001) “Mining web logs for prediction models in WWW caching and prefetching,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 473–478.
- [93] RECKER, M. M. and J. E. PITKOW (1996) “Predicting document access in large multimedia repositories,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, **3**(4), pp. 352–375.
- [94] SMITH, L. and W. WILBUR (2011) “The Popularity of Articles in PubMed,” *Open Information Systems Journal*, **5**, pp. 1–7.
- [95] “Research of tomorrow survey,” <http://www.webarchive.org.uk/wayback/archive/20140614040703/http://www.jisc.ac.uk/publications/reports/2012/researchers-of-tomorrow.aspx>.
- [96] HIGHTOWER, C. and C. CALDWELL (2010) “Shifting sands: science researchers on Google Scholar, Web of Science, and PubMed, with implications for library collections budgets,” *Issues in Science and Technology Librarianship*, (63), p. 4.
- [97] PAGE, L., S. BRIN, R. MOTWANI, and T. WINOGRAD (1999) “The PageRank citation ranking: Bringing order to the web.” .

- [98] SANDERSON, M. and W. B. CROFT (2012) “The history of information retrieval research,” *Proceedings of the IEEE*, **100**(Special Centennial Issue), pp. 1444–1451.
- [99] TAUBE, M., C. GULL, and I. S. WACHTEL (1952) “Unit terms in coordinate indexing,” *American documentation*, **3**(4), pp. 213–218.
- [100] HERSH, W., C. BUCKLEY, T. LEONE, and D. HICKAM (1994) “OHSUMED: An interactive retrieval evaluation and new large test collection for research,” in *SIGIR’94*, Springer, pp. 192–201.
- [101] BRODER, A. (2002) “A taxonomy of web search,” in *ACM Sigir forum*, vol. 36, ACM, pp. 3–10.
- [102] ROSE, D. E. and D. LEVINSON (2004) “Understanding user goals in web search,” in *Proceedings of the 13th international conference on World Wide Web*, ACM, pp. 13–19.
- [103] LI, H., I. COUNCILL, W.-C. LEE, and C. L. 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*, ACM, pp. 883–884.
- [104] TANG, J., J. ZHANG, L. YAO, J. LI, L. ZHANG, and Z. SU (2008) “Arnetminer: extraction and mining of academic social networks,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 990–998.
- [105] TANG, J., J. SUN, C. WANG, and Z. YANG (2009) “Social influence analysis in large-scale networks,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 807–816.
- [106] SUN, Y. and C. L. GILES (2007) “Popularity weighted ranking for academic digital libraries,” in *ECIR*, pp. 605–612.
- [107] ZHOU, D., S. A. ORSHANSKIY, H. ZHA, and C. L. GILES (2007) “Co-ranking Authors and Documents in a Heterogeneous Network,” in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, ICDM ’07*, pp. 739–744.
- [108] AMOLOCHITIS, E., I. T. CHRISTOU, Z.-H. TAN, and R. PRASAD (2013) “A heuristic hierarchical scheme for academic search and retrieval,” *Information Processing & Management*, **49**(6), pp. 1326–1343.

- [109] SOULIER, L., L. BEN JABEUR, L. TAMINE, and W. BAHOUN (2012) “BibRank: a language-based model for co-ranking entities in bibliographic networks,” in *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, ACM, pp. 61–70.
- [110] KANG, I.-H. and G. KIM (2003) “Query type classification for web document retrieval,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ACM, pp. 64–71.
- [111] JANSEN, B. J., D. L. BOOTH, and A. SPINK (2007) “Determining the user intent of web search engine queries,” in *Proceedings of the 16th international conference on World Wide Web*, ACM, pp. 1149–1150.
- [112] LEE, U., Z. LIU, and J. CHO (2005) “Automatic identification of user goals in web search,” in *Proceedings of the 14th international conference on World Wide Web*, ACM, pp. 391–400.
- [113] BALOG, K., Y. FANG, M. DE RIJKE, P. SERDYUKOV, and L. SI (2012) “Expertise retrieval,” *Foundations and Trends in Information Retrieval*, **6**(2–3), pp. 127–256.
- [114] GOLLAPALLI, S. D., P. MITRA, and C. L. GILES (2011) “Ranking authors in digital libraries,” in *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, ACM, pp. 251–254.
- [115] TANG, J., J. ZHANG, R. JIN, Z. YANG, K. CAI, L. ZHANG, and Z. SU (2011) “Topic level expertise search over heterogeneous networks,” *Machine Learning*, **82**(2), pp. 211–237.
- [116] SUN, Y., H. LI, I. G. COUNCILL, J. HUANG, W.-C. LEE, and C. L. GILES (2008) “Personalized Ranking for Digital Libraries Based on Log Analysis,” in *Proceedings of the 10th ACM Workshop on Web Information and Data Management*, WIDM ’08, pp. 133–140.
- [117] HARPALE, A., Y. YANG, S. GOPAL, D. HE, and Z. YUE (2010) “CiteData: A new multi-faceted dataset for evaluating personalized search performance,” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ACM, pp. 549–558.
- [118] ROSEN-ZVI, M., T. GRIFFITHS, M. STEYVERS, and P. SMYTH (2004) “The author-topic model for authors and documents,” in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press, pp. 487–494.

- [119] STEYVERS, M., P. SMYTH, M. ROSEN-ZVI, and T. GRIFFITHS (2004) “Probabilistic author-topic models for information discovery,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 306–315.
- [120] TANG, J., R. JIN, and J. ZHANG (2008) “A Topic Modeling Approach and Its Integration into the Random Walk Framework for Academic Search,” in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM ’08, pp. 1055–1060.
- [121] KHABSA, M. and C. L. GILES (2014) “The number of scholarly documents on the public web,” *PLOS one*, **9**(5), p. e93949.
- [122] CHAWLA, N. V., K. W. BOWYER, L. O. HALL, and W. P. KEGELMEYER (2002) “SMOTE: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, **16**(1), pp. 321–357.
- [123] BAEZA-YATES, R., L. CALDERÓN-BENAVIDES, and C. GONZÁLEZ-CARO (2006) “The intention behind web queries,” in *String processing and information retrieval*, Springer, pp. 98–109.
- [124] LU, Y., F. PENG, X. LI, and N. AHMED (2006) “Coupling feature selection and machine learning methods for navigational query identification,” in *Proceedings of the 15th ACM international conference on Information and knowledge management*, ACM, pp. 682–689.
- [125] LI, X., Y.-Y. WANG, and A. ACERO (2008) “Learning query intent from regularized click graphs,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 339–346.
- [126] DOU, Z., R. SONG, X. YUAN, and J.-R. WEN (2008) “Are click-through data adequate for learning web search rankings?” in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, pp. 73–82.
- [127] MANNING, C. D., P. RAGHAVAN, and H. SCHÜTZE (2008) *Introduction to information retrieval*, vol. 1, Cambridge university press Cambridge.
- [128] ZHAI, C. and J. LAFFERTY (2004) “A study of smoothing methods for language models applied to information retrieval,” *ACM Transactions on Information Systems (TOIS)*, **22**(2), pp. 179–214.
- [129] WU, Q., C. J. BURGESS, K. M. SVORE, and J. GAO (2010) “Adapting boosting for information retrieval measures,” *Information Retrieval*, **13**(3), pp. 254–270.

- [130] BURGESS, C. J. (2010) “From RankNet to LambdaRank to LambdaMART: An Overview,” *MSR-TR-2010-82*.
- [131] BURGESS, C. J. C., R. RAGNO, and Q. V. LE (2006) “Learning to Rank with Nonsmooth Cost Functions,” in *NIPS '06*, pp. 193–200.
- [132] FRIEDMAN, J. H. (2000) “Greedy Function Approximation: A Gradient Boosting Machine,” *Annals of Statistics*, **29**, pp. 1189–1232.

Vita

Madian Khabsa

EDUCATION

The Pennsylvania State University, University Park, PA, USA

Ph.D., Computer Science and Engineering (Graduate minor in Statistics), 2015

M.S., Computer Science and Engineering, 2012

Damascus University, Damascus, Syria

B.S., Information Technology Engineering, 2008

SELECTED PUBLICATIONS

Madian Khabsa and C. Lee Giles. “Chemical entity extraction using CRF and an ensemble of extractors”. *J Cheminform* 7.Suppl 1 (2015): S12.

Madian Khabsa and C. Lee Giles. “The Number of Scholarly Documents on the Public Web.” *PloS one* 9, no. 5 (2014): e93949

Madian Khabsa, Pucktada Treeratpituck, C. Lee Giles. “Large Scale Author Name Disambiguation in Digital Libraries.” *In Big Data (Big Data), 2014 IEEE International Conference on* (pp. 41-42). *IEEE*.

Madian Khabsa, Pucktada Treeratpituck, C. Lee Giles. “AckSeer: A Repository and Search Engine for Automatically Extracted Acknowledgments from Digital Libraries.” *In ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2012*.

Madian Khabsa, Pucktada Treeratpituck, C. Lee Giles. “Entity Resolution using Search Engine Results.” *In ACM International Conference on Information and Knowledge Management (CIKM) 2012*.

EXPERIENCE

University of Chicago. Data Science Fellow, Summer 2014

Qatar Computing Research Institute. Research Associate, Summer 2013

Microsoft Corporation. Software Engineer Intern, Summers of 2011 and 2012