August 2013

# Hierarchical Classification and its Application in University Search

Xiao Li
*The University of Western Ontario*

Supervisor
Charles X. Ling
*The University of Western Ontario*

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

# HIERARCHICAL CLASSIFICATION AND ITS APPLICATION IN UNIVERSITY SEARCH

(Thesis format: Monograph)

by

Xiao Li

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

# Abstract

Web search engines have been adopted by most universities for searching webpages in their own domains. Basically, a user sends keywords to the search engine and the search engine returns a flat ranked list of webpages. However, in university search, user queries are usually related to topics. Simple keyword queries are often insufficient to express topics as keywords. On the other hand, most E-commerce sites allow users to browse and search products in various hierarchies. It would be ideal if hierarchical browsing and keyword search can be seamlessly combined for university search engines. The main difficulty is to automatically classify and rank a massive number of webpages into the topic hierarchies for universities.

In this thesis, we use machine learning and data mining techniques to build a novel hybrid search engine with integrated hierarchies for universities, called SEEU (**S**earch **E**ngine with hi**E**rarchy for **U**niversities).

Firstly, we study the problem of effective hierarchical webpage classification. We develop a parallel webpage classification system based on Support Vector Machines. With extensive experiments on the well-known ODP (Open Directory Project) dataset, we empirically demonstrate that our hierarchical classification system is very effective and outperforms the traditional flat classification approaches significantly.

Secondly, we study the problem of integrating hierarchical classification into the ranking system of keywords-based search engines. We propose a novel ranking framework, called ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), for search engines with hierarchies. Experimental results on four large-scale TREC (Text REtrieval Conference) web search datasets show that our ranking system with hierarchical classification outperforms the traditional flat keywords-based search methods significantly.

Thirdly, we propose a novel active learning framework to improve the performance of hierarchical classification, which is important for ranking webpages in hierarchies. From our experiments on the benchmark text datasets, we find that our active learning framework can achieve good classification performance yet save a considerable number of labeling effort compared with the state-of-the-art active learning methods for hierarchical text classification.

Fourthly, based on the proposed classification and ranking methods, we present a novel hierarchical classification framework for mining academic topics from university webpages. We build an academic topic hierarchy based on the commonly accepted Wikipedia academic disciplines. Based on this hierarchy, we train a hierarchical classifier and apply it to mine academic topics. According to our comprehensive analysis, the academic topics mined by our method are reasonable and consistent with the real-world topic distribution in universities.

Finally, we combine all the proposed techniques together and implement the SEEU search engine. According to two usability studies conducted in the ECE and the CS departments at our university, SEEU is favored by the majority of participants.

To conclude, the main contribution of this thesis is a novel search engine, called SEEU, for universities. We discuss the challenges toward building SEEU and propose effective machine learning and data mining methods to tackle them. With extensive experiments on well-known benchmark datasets and real-world university webpage datasets, we demonstrate that our system is very effective. In addition, two usability studies of SEEU in our university show that SEEU has a great promise for university search.

**Keywords:** Hierarchical Classification, Search Engine, System Evaluation

# Acknowledgements

First of all, I would like to thank my supervisor, Dr. Charles Ling, for his guidance, encouragement and support throughout my PhD study. Without his instruction, I could not have finished this thesis.

I would thank my thesis committee members, Dr. Sylvia Osborn, Dr. John Barron, Dr. Luiz Fernando Capretz and Dr. Dan Wu, who graciously agreed to serve on my committee.

Thanks are also given to other members of Data Mining Lab at Western, Jun Du, Eileen Ni, Da Kuang, Yan Luo, David DeAngelis, Arman Didandeh, Nima Mirbakhsh and Shuang Ao, for their collaboration and valuable discussions. Many thanks to Da Kuang. We cooperated on the early version of SEEU search engine and jointly published two papers. I also want to thank Nima Mirbakhsh for designing the logo of SEEU.

I would thank Dr. Huaimin Wang, the advisor for my Master study at National University of Defense Technology. He guided me in the research of distributed computing and data mining and recommended me to conduct doctoral study at Western.

I would like to thank Dr. Ali Bou Nassif from the ECE department of Western for coordinating the usability study of SEEU.

Finally, my gratitude to my parents, for their support and encouragement in pursuing the PhD degree in Western. This dissertation would not have been possible without their support.

My work has been financially supported by different sources, including China Scholarship Council, NSERC and National High Technology Research and Development Program of China (Grant No. 2012AA011201).

# Contents

# List of Figures

# List of Tables

# List of Appendices

# Chapter 1

# Introduction

Since the emergence of WWW techniques in the 1990s, the web has become the major informational medium for both faculty and students in universities. Every day, thousands of education and research activities are published as e-documents on the web. With the explosive growth of university webpages published online, effective organization and search of university webpages is very important.

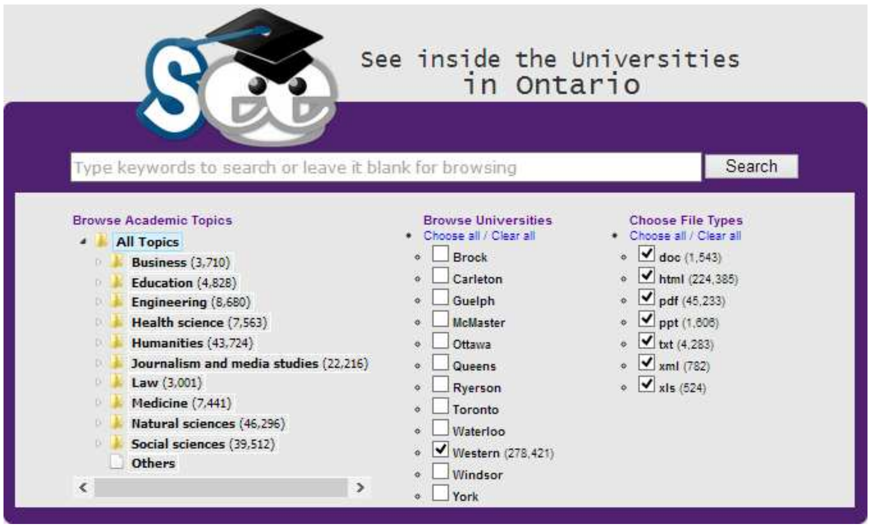
Web search engines, due to their great success on the general web, have been adopted by many universities for searching webpages in their own domains. Most of the existing web search engines (such as Google) licensed by universities are only based on keyword search. Basically, a user sends keywords to a search engine and the search engine returns a flat ranked list of webpages.

However, searching webpages in universities is different from searching information on the general web. In a university search, user queries are often related to topics (e.g., academics, campus life, news and media) [16]. Simple keyword queries are often insufficient to express complex topics as keywords [29] (e.g., finding professors in a specific research area among multiple universities). On the other hand, modern faceted search engines [51] in E-commerce sites (such as Amazon and eBay) let users browse and search in the categories of various hierarchies (such as product categories and regions of the world). Such faceted search engines have been shown to be more effective for searching information in complex hierarchies than general keywords-based search engines [50].

It would be ideal if hierarchical browsing and keyword search can be seamlessly combined for university search engines. Such hybrid search engines allow users to search by keywords as in Google while drilling down to any (sub)category in multiple hierarchies. If users do not choose any hierarchy, it would be the same as the current Google. On the other hand, users can also browse any (sub)category without keywords, and the search engine will return a list of the most popular webpages (e.g., the webpages with the highest PageRank values [85]) within that category.

In this thesis, we propose a novel hybrid search engine, called SEEU (**S**earch **E**ngine with hi**E**rarchy for **U**niversities), for Canadian universities to facilitate research collaboration, and help faculty members and students find desired webpages more easily. The novel and key idea is to incorporate multiple relevant hierarchies (such as academic topics, universities and media types) for university webpages in SEEU. See Figure 1.1 (the home page) and Figure 1.2

(the result page) for the user interface (UI) of SEEU[1]. In SEEU, about two million webpages from the top 12 largest Canadian universities are crawled, processed, and then classified into hierarchies.



Type keywords to search or leave it blank for browsing

Browse Academic Topics
- All Topics
  - Business (3,710)
  - Education (4,828)
  - Engineering (8,680)
  - Health science (7,563)
  - Humanities (43,724)
  - Journalism and media studies (22,216)
  - Law (3,001)
  - Medicine (7,441)
  - Natural sciences (46,296)
  - Social sciences (39,512)
  - Others

Browse Universities
- Choose all / Clear all
- ☐ Brock
- ☐ Carleton
- ☐ Guelph
- ☐ McMaster
- ☐ Ottawa
- ☐ Queens
- ☐ Ryerson
- ☐ Toronto
- ☐ Waterloo
- ☑ Western (278,421)
- ☐ Windsor
- ☐ York

Choose File Types
- Choose all / Clear all
- ☑ doc (1,543)
- ☑ html (224,385)
- ☑ pdf (45,233)
- ☑ ppt (1,606)
- ☑ txt (4,283)
- ☑ xml (782)
- ☑ xls (524)

Figure 1.1: The home page of SEEU. Beneath the search box are the topic hierarchy, the university hierarchy and the file type hierarchy.

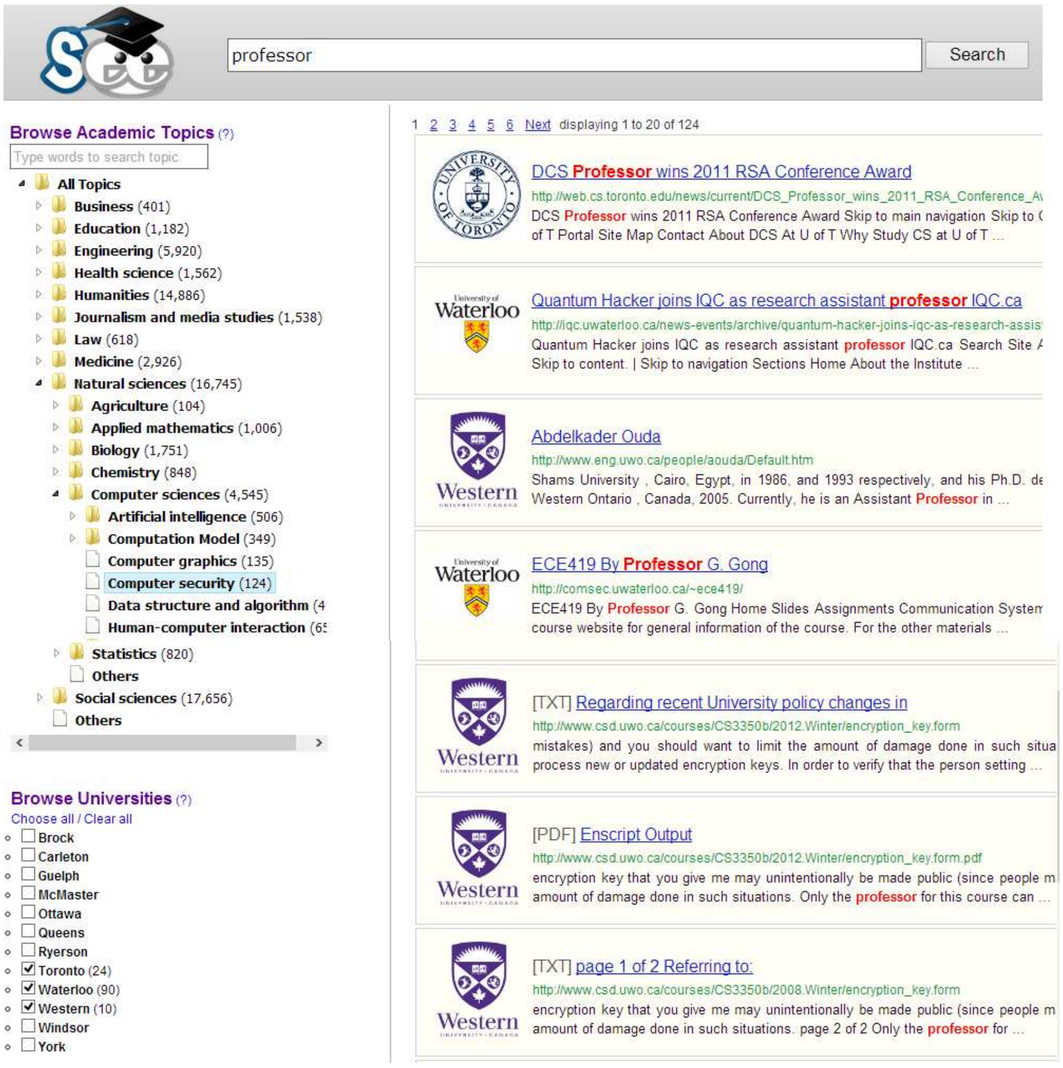
The main challenge in building SEEU is to define commonly accepted hierarchies, and automatically classify and rank a massive number of webpages into various hierarchies (such as academics, campus life and media types) for universities. Although hierarchical faceted search engines [50] have already implemented such functionality, it should be noted that the items (e.g, products, books, CDs and so on) in these search engines are pre-labeled by human experts. Give the large number of webpages published daily in universities, manual labeling is not feasible.

In this thesis, we use machine learning and data mining methods to tackle these challenges. Firstly, we develop an effective hierarchical webpage classification system for large-scale webpage categorization in SEEU. Secondly, we propose the ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), a novel ranking framework that combines hierarchical classification with keywords-based ranking. Thirdly, we propose a novel active learning framework to improve hierarchical classification, which is very important for ranking in hierarchies. With extensive experiments on well-known benchmark classification and web search datasets, we empirically demonstrate that our proposed methods are very effective and they outperform the traditional flat classification and search methods significantly.

Based on these techniques, we propose a new hierarchical classification framework to mine academic topics from the challenging two million university webpages in SEEU. Specifically, we build the academic topic hierarchy based on the commonly accepted Wikipedia academic disciplines. We train a hierarchical classifier and apply it to classify university webpages into the academic topic hierarchy. According to our comprehensive analysis, the academic topic pattern mined by our system is reasonable, consistent with the real-world topic distribution in

[1]SEEU can be visited at `http://kdd.csd.uwo.ca:88/seeu`

most universities, and better than the state-of-the-art topic modeling methods.



professor
Search

Browse Academic Topics (?)
Type words to search topic
- All Topics
  - Business (401)
  - Education (1,182)
  - Engineering (5,920)
  - Health science (1,562)
  - Humanities (14,886)
  - Journalism and media studies (1,538)
  - Law (618)
  - Medicine (2,926)
  - Natural sciences (16,745)
    - Agriculture (104)
    - Applied mathematics (1,006)
    - Biology (1,751)
    - Chemistry (848)
    - Computer sciences (4,545)
      - Artificial intelligence (506)
      - Computation Model (349)
      - Computer graphics (135)
      - Computer security (124)
      - Data structure and algorithm (4
      - Human-computer interaction (65
    - Statistics (820)
    - Others
  - Social sciences (17,656)
  - Others

Browse Universities (?)
Choose all / Clear all
- ☐ Brock
- ☐ Carleton
- ☐ Guelph
- ☐ McMaster
- ☐ Ottawa
- ☐ Queens
- ☐ Ryerson
- ☑ Toronto (24)
- ☑ Waterloo (90)
- ☑ Western (10)
- ☐ Windsor
- ☐ York

1 2 3 4 5 6 Next displaying 1 to 20 of 124

DCS Professor wins 2011 RSA Conference Award
http://web.cs.toronto.edu/news/current/DCS_Professor_wins_2011_RSA_Conference_A
DCS Professor wins 2011 RSA Conference Award Skip to main navigation Skip to
of T Portal Site Map Contact About DCS At U of T Why Study CS at U of T ...

Quantum Hacker joins IQC as research assistant professor IQC.ca
http://iqc.uwaterloo.ca/news-events/archive/quantum-hacker-joins-iqc-as-research-assis
Quantum Hacker joins IQC as research assistant professor IQC.ca Search Site
Skip to content. | Skip to navigation Sections Home About the Institute ...

Abdelkader Ouda
http://www.eng.uwo.ca/people/aouda/Default.htm
Shams University , Cairo, Egypt, in 1986, and 1993 respectively, and his Ph.D. de
Western Ontario , Canada, 2005. Currently, he is an Assistant Professor in ...

ECE419 By Professor G. Gong
http://comsec.uwaterloo.ca/~ece419/
ECE419 By Professor G. Gong Home Slides Assignments Communication Syster
course website for general information of the course. For the other materials ...

[TXT] Regarding recent University policy changes in
http://www.csd.uwo.ca/courses/CS3350b/2012.Winter/encryption_key.form
mistakes) and you should want to limit the amount of damage done in such situa
process new or updated encryption keys. In order to verify that the person setting ...

[PDF] Enscript Output
http://www.csd.uwo.ca/courses/CS3350b/2012.Winter/encryption_key.form.pdf
encryption key that you give me may unintentionally be made public (since people m
amount of damage done in such situations. Only the professor for this course can ...

[TXT] page 1 of 2 Referring to:
http://www.csd.uwo.ca/courses/CS3350b/2008.Winter/encryption_key.form
encryption key that you give me may unintentionally be made public (since people m
amount of damage done in such situations. page 2 of 2 Only the professor for ...

Figure 1.2: The result page of SEEU. It shows an example of searching for "professor" in the "Computer Security" category in Toronto, Waterloo and Western in SEEU.

Finally, we combine all the proposed techniques together and implement the SEEU search engine. According to two usability studies conducted in the ECE and the CS departments at our university, SEEU is favored by the majority of participants.

The rest of this chapter is organized as follows. In Section 1.1, we review several search engine approaches that are closely related to ours. They are keywords-based search engines, web directory and faceted search engines. In Section 1.2, we analyze the advantages and disadvantages of these approaches when applied for university search. In Section 1.3, we describe the motivation and challenges for a new search engine for universities. We present the user interface of SEEU, and briefly discuss the advantages and implementation challenges of SEEU. In Section 1.4, we list our major contributions in this thesis. Finally, we close this chapter by presenting the thesis outline in the last section.

## 1.1 Search Engine Approach Review

In this section, we review three search engine approaches that are closely related to our works. They are keywords-based search engines, web directory and faceted search engines. We describe the basic concept and technique behind these search engines, and discuss their advantages and disadvantages.

### 1.1.1 keywords-based Search Engines

The first search engine approach we discuss is the keywords-based search. keywords-based search engines are currently the most popular search engine approach for the web. They provide users a simple and intuitive way to find information. For example, Figure 1.3 shows the user interface of the Google search engine. It simply contains a search box and several search buttons. When users type keywords in the search box and click the search button, Google will quickly return a ranked list of relevant results (in several hundred milliseconds).

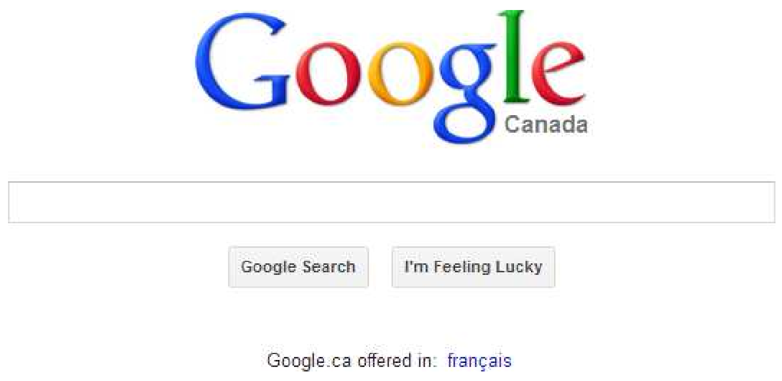


Figure 1.3: The home page of Google search engine.

How can the keywords-based search engines, such as Google, return results so quickly? To achieve this, keywords-based search engines usually rely on three important components to effectively rank the massive number of webpages. They are web crawlers, index databases and searching model.

The task of web crawlers (or web spiders, web robots) is to crawl webpages and store useful data into the index database. Specifically, keywords-based search engines usually launch many web crawlers that periodically surf the web. Each web crawler fetches webpage content from remote web servers, extract hyperlinks as well as text data in webpages and stores them into the index database (explained later). After that, the web crawler follows the hyperlinks on the webpage to fetch more webpages.

The index database is the key to efficiently retrieve the relevant webpages in a short period. Given a user query, a naive approach to search is to scan the crawled webpages sequentially. However, it fails when the crawled dataset is very large, such as webpages on the entire web. To deal with this challenge, keywords-based search engines often use a data structure, called inverted indices [3], to speed up the search. Simply speaking, for a text collection, the inverted indices maintain a table that maps each word to a list of document-position pairs. For example, given a text collection crawled by web crawlers,

- $d_1$ =“Active Learning for Hierarchical Text Classification”.
- $d_2$ =“A Survey of Hierarchical Text Classification”.
- $d_3$ =“Engaging Students Through Active Learning”.

The inverted indices are incrementally built when the search engine indexes the three documents. The results are shown in Table 1.1. It contains multiple document-position lists. Each list contains all the word occurrence for a unique word in the data collection. For example, for the word “active”, $\{(1,1),(3,4)\}$ means that the word “active” appears at the first position in $d_1$ and the fourth position in $d_3$.

Table 1.1: An example of inverted indices. The two numbers inside each bracket are the document ID and the word position.

| Word | Document-position list |
|---|---|
| “active” | $\{(1,1),(3,4)\}$ |
| “learning” | $\{(1,2),(3,5)\}$ |
| “for” | $\{(1,3)\}$ |
| “hierarchical” | $\{(1,4),(2,4)\}$ |
| “text” | $\{(1,5),(2,5)\}$ |
| “classification” | $\{(1,6),(2,6)\}$ |
| “a” | $\{(2,1)\}$ |
| “survey” | $\{(2,2)\}$ |
| “of” | $\{(2,3)\}$ |
| “engaging” | $\{(3,1)\}$ |
| “students” | $\{(3,2)\}$ |
| “through” | $\{(3,3)\}$ |

Given a query “active learning”, the search engine extracts the words “active” and “learning” from the query, conducts set intersection on the document IDs of their corresponding document-position lists, and returns two documents, $d_1$ and $d_3$, to the user. The positions stored in the database can be further used to highlight the keywords in results.

It has been shown that both the space complexity and time complexity of searching through inverted indices are close to $O(n^{0.85})$ [3]. Thus, the inverted indices allow search engines to find relevant webpages very quickly.

The third component of keywords-based search engine is the searching model. Many searching models have been proposed for keywords-based search engines. The simplest searching model is the *boolean model* [43] which is based on boolean algebra. In the *boolean model*, the occurrences of words in both queries and documents are assigned binary truth values (i.e., *true* or *false*). Users familiar with boolean logic can use boolean operators (e.g., AND, OR, NOT) to connect keywords, and thus express complex search intention. The search engine only returns relevant results satisfying the boolean expressions formed by users with *true* value. However, a drawback of the *boolean model* is that the results are judged based on a binary decision value (i.e., *true* or *false*). As the *boolean model* can not provide graded scores, it is impossible to further rank the relevant results. In addition, for normal users, it is often difficult to translate search intention precisely into boolean expressions.

To resolve the binary matching problem in the *boolean model*, we can resort to the *vector space model* [99, 100]. The basic idea of the *vector space model* is to represent both queries and documents as vectors of words, and assign a numerical weight, not binary (true or false) weight, to each word. Those word vectors can be used to compute the *degree of similarity* between indexed documents and the query. Thus, the *vector space model* can measure the relevance of documents to the query much more precisely. For example, given a query vector $v_q = (w_1, w_2, \ldots, w_n)$ and a document vector $v_d = (w_1, w_2, \ldots, w_n)$[2], we can measure their similarity by cosine similarity, which is

$$sim(v_q, v_d) = \frac{v_q \cdot v_d}{||v_q|| \cdot ||v_d||} \tag{1.1}$$

where $v_q \cdot v_d$ is the *inner product* of $v_q$ and $v_d$; $||v_q||$ is the *norm* of $v_q$ and $||v_d||$ is the *norm* of $v_d$..

The most popular weighting schema in the *vector space model* is the TF·IDF weighting. It measures both the term frequency (TF) and the inverse document frequency (IDF) for a matched word in the query. The TF factor assigns high weights to the frequent words in documents. Usually, the more often a word appears in the document, the more relevant it is against the query. For the IDF factor, the intuitive idea is that common words occurring in many documents are not discriminative to distinguish a relevant document from irrelevant ones.

Formally, given a document $d$, the TF·IDF score of a term $t$ is computed as

$$TFIDF(t, d) = TF(t, d) \cdot IDF(t, D) \tag{1.2}$$

where

$$TF(t, d) = \frac{f(t, d)}{\sum_{t \in d} f(t, d)} \quad \text{and} \quad IDF(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|}$$

$f(t, d)$ is the number of occurrences of term $t$ in document $d$ and $|D|$ is the total number of documents in the collection.

The main drawback of the *vector space model* is that it can not deal with synonymy (i.e., multiple words that have similar meanings) and polysemy (i.e., a word may have multiple senses and multiple types of usage in different contexts) because the *vector space model* needs to exactly match terms in queries and documents.

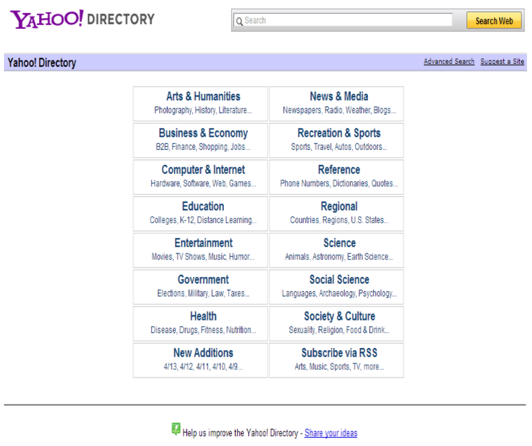
### 1.1.2 Web Directory

The web directory is a directory service for the web. It organizes websites (or individual webpages) into a *predefined* topic hierarchy (taxonomy). The taxonomy can be either a general purpose topic hierarchy, or a hierarchy related to a specific area. Examples of well known general web directories are the Yahoo! Directory[3] and the Open Directory Project (ODP)[4]. Figure 1.4 shows their user interfaces. We can see that unlike the keywords-based search engines, a web directory contains a topic hierarchy on the home page. Users can browse webpages in categories of the hierarchy without typing keywords.

[2]For the two vectors, $w_i = 1$ means that the word $w_i$ exists in the query or the document and $w_i = 0$ means it does not exist.

[3]`http://dir.yahoo.com/`

[4]`http://www.dmoz.org`



YAHOO! DIRECTORY

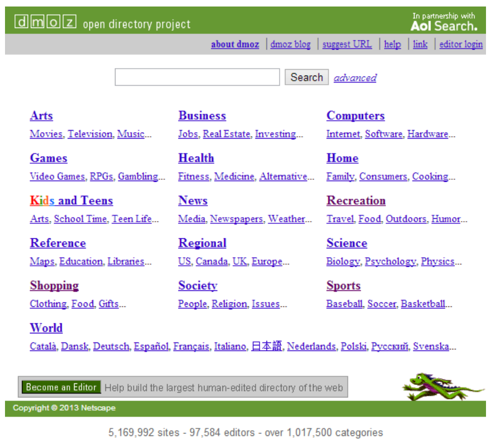
Search | Search Web

Yahoo! Directory — Advanced Search  Suggest a Site

| | |
|---|---|
| Arts & Humanities<br>Photography, History, Literature... | News & Media<br>Newspapers, Radio, Weather, Blogs... |
| Business & Economy<br>B2B, Finance, Shopping, Jobs... | Recreation & Sports<br>Sports, Travel, Autos, Outdoors... |
| Computer & Internet<br>Hardware, Software, Web, Games... | Reference<br>Phone Numbers, Dictionaries, Quotes... |
| Education<br>Colleges, K-12, Distance Learning... | Regional<br>Countries, Regions, U.S. States... |
| Entertainment<br>Movies, TV Shows, Music, Humor... | Science<br>Animals, Astronomy, Earth Science... |
| Government<br>Elections, Military, Law, Taxes... | Social Science<br>Languages, Archaeology, Psychology... |
| Health<br>Disease, Drugs, Fitness, Nutrition... | Society & Culture<br>Sexuality, Religion, Food & Drink... |
| New Additions<br>4/13, 4/12, 4/11, 4/10, 4/9... | Subscribe via RSS<br>Arts, Music, Sports, TV, more... |

Help us improve the Yahoo! Directory - Share your ideas

(a) Yahoo! Directory



dmoz open directory project — In partnership with Aol Search.

about dmoz | dmoz blog | suggest URL | help | link | editor login

Search  advanced

| | | |
|---|---|---|
| Arts<br>Movies, Television, Music... | Business<br>Jobs, Real Estate, Investing... | Computers<br>Internet, Software, Hardware... |
| Games<br>Video Games, RPGs, Gambling... | Health<br>Fitness, Medicine, Alternative... | Home<br>Family, Consumers, Cooking... |
| Kids and Teens<br>Arts, School Time, Teen Life... | News<br>Media, Newspapers, Weather... | Recreation<br>Travel, Food, Outdoors, Humor... |
| Reference<br>Maps, Education, Libraries... | Regional<br>US, Canada, UK, Europe... | Science<br>Biology, Psychology, Physics... |
| Shopping<br>Clothing, Food, Gifts... | Society<br>People, Religion, Issues... | Sports<br>Baseball, Soccer, Basketball... |

World
Català, Dansk, Deutsch, Español, Français, Italiano, 日本語, Nederlands, Polski, Русский, Svenska...

Become an Editor  Help build the largest human-edited directory of the web

Copyright © 2013 Netscape

5,169,992 sites - 97,584 editors - over 1,017,500 categories

(b) ODP

Figure 1.4: The home pages of (a) Yahoo! Directory and (b) ODP web directory.

We give a brief discussion of the Yahoo! Directory and the ODP directory. The original Yahoo! search engine is the oldest example of a web directory. It relies on human efforts to organize webpages into hierarchies. Usually, the editing work is conducted by paid employees in Yahoo!. Due to the slow editing process by a small group of paid workers, the Yahoo! Directory can not keep pace with the fast growth of the web. On the other hand, ODP applies an open and free editing policy. Any Internet user can apply to be an editor of a category in the hierarchy. When an anonymous user submits a website to a category for inclusion, the corresponding editors review the submission request and decide to accept it or not. Due to the open editorial policy, ODP has become the largest web directory on the web. From the statistics shown on the ODP home page in May 2013, there are over 98,000 editors working on over one million categories that include 5.1 million webpages.[5]

The major advantage of a web directory over keywords-based search engines is the hierarchical browsing functionality. Users interested in specific topics can browse webpages in corresponding categories without typing keywords. In addition, as webpages in web directories are categorized by human editors, the quality of webpage classification in a taxonomy is usually very high. Thus, the relevance of webpages in each category is often satisfying. However, a drawback of a web directory is that the relevant webpages in each category are very limited. This is because manual classification is too slow to classify all the webpages on the web. For this reason, web directories mainly acts as an archive (Yahoo! Directory) or data providers (e.g., ODP) for Internet companies, such as AOL, Netscape Search and Google Directory.

## 1.1.3 Faceted Search Engines

The faceted search engine is a hybrid search engine approach that combines keyword search and hierarchical browsing (or web directory) to support exploration and discovery within an information collection [51]. The usefulness of faceted search has been demonstrated by its numerous applications in E-commerce sites (such as Amazon and e-Bay) [111]. The key difference that distinguishes faceted search engines from general keywords-based search engines

[5]See the statistics at the ODP website `http://www.dmoz.org`.

is a set of flat or hierarchical facets (see the left panel of the user interface at Amazon in Figure 1.5). These facets can be seen as hierarchies that organize items (e.g., products, documents, images and so on). When users select labels of facets in the user interface (e.g., selecting "Movies & TV" under the facet "Departments"), the faceted search engine returns results that satisfy the conjunctive normal form (logical conjunction of disjunctions) over the selected labels under each facet[51].

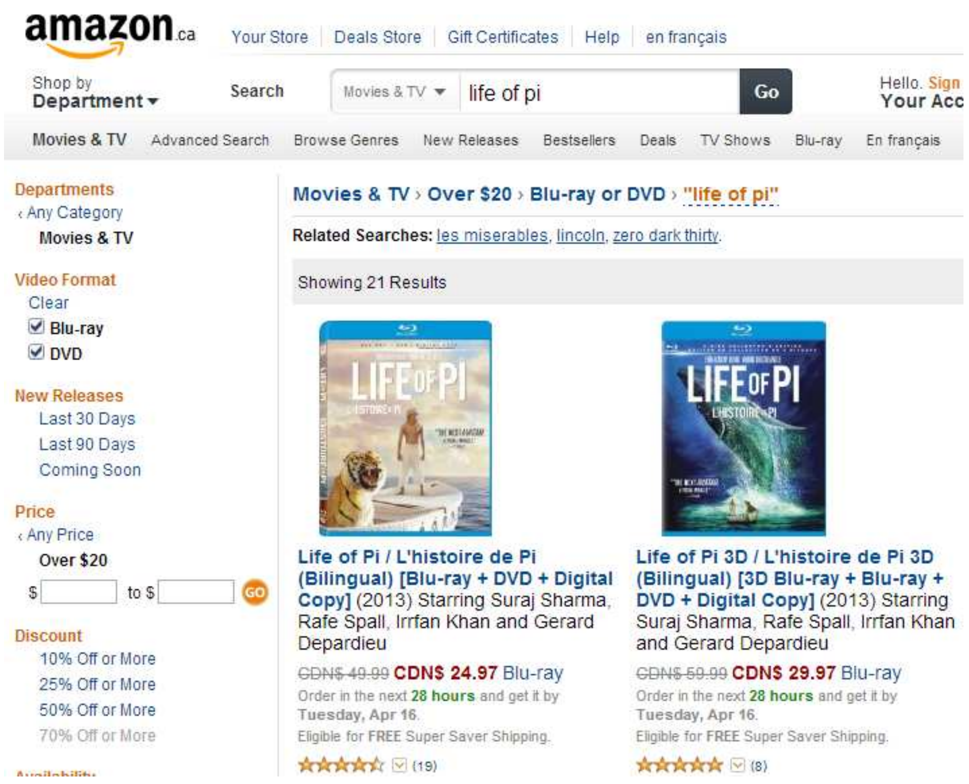
For example, Figure 1.5 shows an example of searching the movie "Life of Pi" on Amazon. A user chooses the facet labels as

- Departments: "Movies & TV"
- Video Format: "Blue-ray" or "DVD"
- Price: "Over $20"

These facet values are translated into a conjunctive normal form as

$$\begin{aligned} Query \;=\; & (Departments = \text{"Movies \& TV"}) \\ & \wedge (VideoFormat = \text{"Blue-ray"} \vee VideoFormat = \text{"DVD"}) \\ & \wedge (Price \geq 20) \end{aligned}$$

where $\wedge$ is the operator of logic conjunction and $\vee$ is the operator of logic disjunction.



Movies & TV › Over $20 › Blu-ray or DVD › "life of pi"

Related Searches: les miserables, lincoln, zero dark thirty.

Showing 21 Results

Life of Pi / L'histoire de Pi (Bilingual) [Blu-ray + DVD + Digital Copy] (2013) Starring Suraj Sharma, Rafe Spall, Irrfan Khan and Gerard Depardieu
~~CDN$ 49.99~~ CDN$ 24.97 Blu-ray
Order in the next 28 hours and get it by Tuesday, Apr 16.
Eligible for FREE Super Saver Shipping.
(19)

Life of Pi 3D / L'histoire de Pi 3D (Bilingual) [3D Blu-ray + Blu-ray + DVD + Digital Copy] (2013) Starring Suraj Sharma, Rafe Spall, Irrfan Khan and Gerard Depardieu
~~CDN$ 59.99~~ CDN$ 29.97 Blu-ray
Order in the next 28 hours and get it by Tuesday, Apr 16.
Eligible for FREE Super Saver Shipping.
(8)

Figure 1.5: An example of searching for the 2012 movie "Life of Pi" on the Amazon site.

Due to the integrated facet hierarchies, faceted search engines have several advantages over the keywords-based search engines and the web directories. Firstly, when users issue complex search requests (such as the searching "Life of Pi" example above), using faceted search engines is more intuitive and natural than the keywords-based search engines. Users do not need to precisely formalize their search intention as a complex boolean expression. They can simply click the corresponding facet labels. Secondly, different from the web directories which only have one topic hierarchy, faceted search engines have multiple orthogonal facet hierarchies for users to refine results. This is very helpful to search products (such as books, movies and laptops) with different meta properties (i.e., brands, price, size, releasing date, authors and so on).

We can see that the key to the success of faceted search engines is the facet hierarchies and the classification of items into these hierarchies [6]. In most E-commerce sites (such as Amazon), as their product databases already provide rich structured meta properties (such as price, brands and size), both the hierarchies and the classification of items can be extracted without much effort. However, in unstructured data collections, such as a webpage dataset, it could be very challenging. Some authors [30, 108, 72] have proposed methods to automatically building facet hierarchies and classification of items. Although these methods do not need human supervision, the hierarchies generated by these methods heavily depend on the quality of corpora [30]. For webpage collections that contain a lot of heterogeneous data, the quality of the hierarchies generated by these methods may not be satisfying.

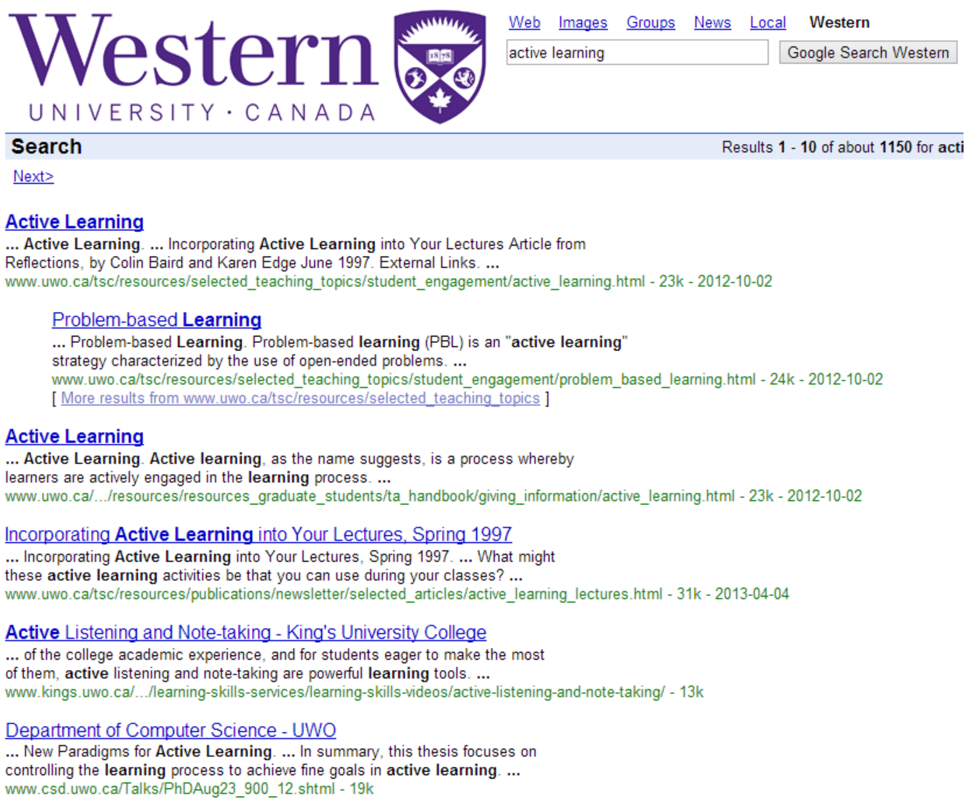
In this section, we have discussed three different search engine approaches. These approaches have many successful applications on the web. However, when we apply these approaches to search university webpages, will we always get satisfying results? In the next section, we analyze the limitation of these approaches when applied in university search.

## 1.2 Limitation of Current Approaches

Many Canadian universities currently license Google's search with "site:" to restrict search results to be within their own domains[6]. Thus, we will firstly analyze the limitation of using keywords-based search engines, such as Google, for searching university webpages.

When professors and graduate students use keywords-based search engines to search university webpages, the queries sent by them are usually related to academic topics. However, simple keywords-based search is often very limited in expressing topics as keywords [29]. For example, consider searching for "active learning", a research field in Computer Science (CS). When a graduate student in CS department sends the keywords to a keywords-based search engine, such as Google, she expects the search engine to return the results that not only contain keywords "active learning" but also belong to CS research. However, the returned results (see Figure 1.6) are actually quite noisy. From Figure 1.6, we can see that only the 6th result is about CS research. The other top five results belong to education research. This is because "active learning" can also mean a methodology in education research that increases student engagement in classrooms [90].

[6]The *Google Search Western* at the home page of Western is essentially the Google search with "site:uwo.ca".


Western UNIVERSITY · CANADA
Web Images Groups News Local Western
active learning
Google Search Western

Search Results 1 - 10 of about 1150 for acti

Next>

Active Learning
... Active Learning. ... Incorporating Active Learning into Your Lectures Article from Reflections, by Colin Baird and Karen Edge June 1997. External Links. ...
www.uwo.ca/tsc/resources/selected_teaching_topics/student_engagement/active_learning.html - 23k - 2012-10-02

Problem-based Learning
... Problem-based Learning. Problem-based learning (PBL) is an "active learning" strategy characterized by the use of open-ended problems. ...
www.uwo.ca/tsc/resources/selected_teaching_topics/student_engagement/problem_based_learning.html - 24k - 2012-10-02
[ More results from www.uwo.ca/tsc/resources/selected_teaching_topics ]

Active Learning
... Active Learning. Active learning, as the name suggests, is a process whereby learners are actively engaged in the learning process. ...
www.uwo.ca/.../resources/resources_graduate_students/ta_handbook/giving_information/active_learning.html - 23k - 2012-10-02

Incorporating Active Learning into Your Lectures, Spring 1997
... Incorporating Active Learning into Your Lectures, Spring 1997. ... What might these active learning activities be that you can use during your classes? ...
www.uwo.ca/tsc/resources/publications/newsletter/selected_articles/active_learning_lectures.html - 31k - 2013-04-04

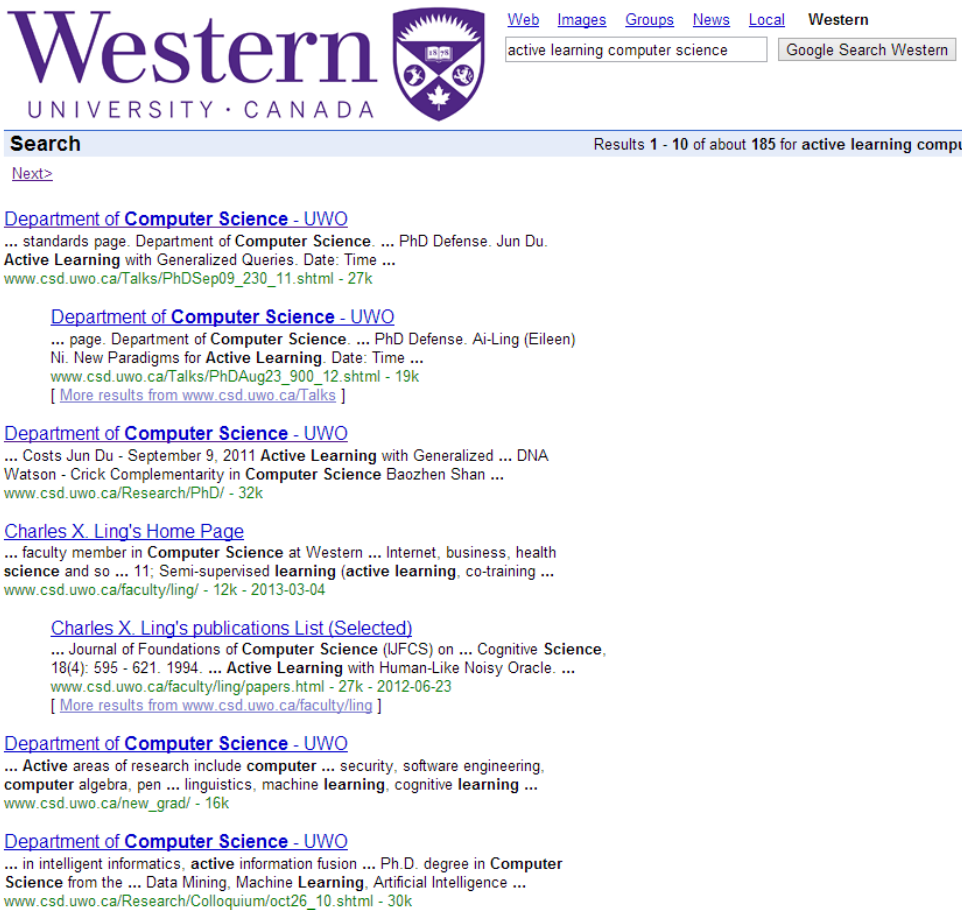
Active Listening and Note-taking - King's University College
... of the college academic experience, and for students eager to make the most of them, active listening and note-taking are powerful learning tools. ...
www.kings.uwo.ca/.../learning-skills-services/learning-skills-videos/active-listening-and-note-taking/ - 13k

Department of Computer Science - UWO
... New Paradigms for Active Learning. ... In summary, this thesis focuses on controlling the learning process to achieve fine goals in active learning. ...
www.csd.uwo.ca/Talks/PhDAug23_900_12.shtml - 19k

Figure 1.6: Search "active learning", a research field in Computer Science in *Google Search Western*.

One may say that for this example we can type more keywords to refine results, such as "active learning computer science". We try this query and show the results in Figure 1.7. We can see that although all the results are related to CS research, each result must contain the keywords "computer science". This is also problematic. Because some important webpages without the keywords "computer science" will now be filtered out. If we type more keywords, the results will become even worse as more results will be filtered out.

To solve the keyword ambiguity problem, we may resort to the web directory approach where people can browse webpages without typing any keywords. Suppose there exists a very large topic hierarchy that covers all the academic topics. For the above case, a user can browse the category "active learning" in the hierarchy and may quickly find the relevant results. Although this approach sounds perfect to solve the keyword ambiguity problem, it is unrealistic for several reasons. Firstly, it is quite difficult to define a commonly accepted hierarchy to capture all the academic topics. If such a hierarchy exists, it could be very large and deep, and thus inconvenient for users to use. Secondly, it is ineffective to only browse one topic hierarchy to find the desired results. For example, suppose we want to find "Computer Science" related textbooks. If we just browse the webpages under the category "Computer Science", we may need to sequentially scan many pages of results to find related textbooks. The larger a category is, the slower the result scanning will be.


Western
UNIVERSITY · CANADA
Web Images Groups News Local Western
active learning computer science
Google Search Western

Search Results 1 - 10 of about 185 for active learning compu

Next>

Department of Computer Science - UWO
... standards page. Department of Computer Science. ... PhD Defense. Jun Du. Active Learning with Generalized Queries. Date: Time ...
www.csd.uwo.ca/Talks/PhDSep09_230_11.shtml - 27k

Department of Computer Science - UWO
... page. Department of Computer Science. ... PhD Defense. Ai-Ling (Eileen) Ni. New Paradigms for Active Learning. Date: Time ...
www.csd.uwo.ca/Talks/PhDAug23_900_12.shtml - 19k
[ More results from www.csd.uwo.ca/Talks ]

Department of Computer Science - UWO
... Costs Jun Du - September 9, 2011 Active Learning with Generalized ... DNA Watson - Crick Complementarity in Computer Science Baozhen Shan ...
www.csd.uwo.ca/Research/PhD/ - 32k

Charles X. Ling's Home Page
... faculty member in Computer Science at Western ... Internet, business, health science and so ... 11; Semi-supervised learning (active learning, co-training ...
www.csd.uwo.ca/faculty/ling/ - 12k - 2013-03-04

Charles X. Ling's publications List (Selected)
... Journal of Foundations of Computer Science (IJFCS) on ... Cognitive Science, 18(4): 595 - 621. 1994. ... Active Learning with Human-Like Noisy Oracle. ...
www.csd.uwo.ca/faculty/ling/papers.html - 27k - 2012-06-23
[ More results from www.csd.uwo.ca/faculty/ling ]

Department of Computer Science - UWO
... Active areas of research include computer ... security, software engineering, computer algebra, pen ... linguistics, machine learning, cognitive learning ...
www.csd.uwo.ca/new_grad/ - 16k

Department of Computer Science - UWO
... in intelligent informatics, active information fusion ... Ph.D. degree in Computer Science from the ... Data Mining, Machine Learning, Artificial Intelligence ...
www.csd.uwo.ca/Research/Colloquium/oct26_10.shtml - 30k

Figure 1.7: Search "active learning computer science" in *Google Search Western*.

In fact, we can combine the keywords-based approach and the web directory approach together to better resolve the keyword ambiguity problem. This is actually the faceted search engine approach. It inherits the advantages of both approaches. Faceted search engines allow users to search by keywords as in a keywords-based search engine while choosing (i.e., drilling down) any (sub)category in the hierarchy to refine the results solely in that category. For example, to find "Computer Science" related textbooks, we can search keywords "textbooks" in the category "Computer Science".

However, faceted search engine approaches still have the bottleneck of effective classification of webpages into hierarchies. Different to the E-commerce sites where products have rich pre-labeled meta properties (such as price, brands and size), normal webpages do not provide explicit information about topics. One may say that similar to the web directory approach, we could recruit professionals (i.e., professors and research assistants) in universities to generate a high quality classification of webpages. However, it should be noted that human classification speed is quite slow compared with the explosive growth rate of webpages. For Western, the total number of webpages already exceeds two million (as estimated by Google), and this number is quickly growing every day.

To summarize, in this section, we have analyzed the limitations of applying current search engine approaches for searching universities webpages. One of the major difficulties is to solve the keyword ambiguity problem. Although these approaches can be adopted to tackle this and other problems, they are either based on unrealistic assumptions or lack of effective algorithms. In the next section, we propose a new search engine for university search.

## 1.3 SEEU: A New Search Engine with Integrated Hierarchies for Universities

In this thesis, we propose a novel search engine with integrated hierarchies, called SEEU[7] (**S**earch **E**ngine with hi**E**rarchies for **U**niversities), for Canadian universities to facilitate research collaboration, and help faculty members and students find desired webpages more easily. The novelty of SEEU is to incorporate multiple relevant hierarchies (such as academic topics, universities and media types) for university webpages in SEEU. See Figure 1.1 (home page) and Figure 1.2 (result page) for the user interface (UI) of SEEU. In SEEU, about two million webpages from the top 12 largest Canadian universities are crawled, processed, and then classified (using data mining and machine learning methods) into hierarchies based on their content, not the department or faculty structure.

### 1.3.1 Advantages of SEEU

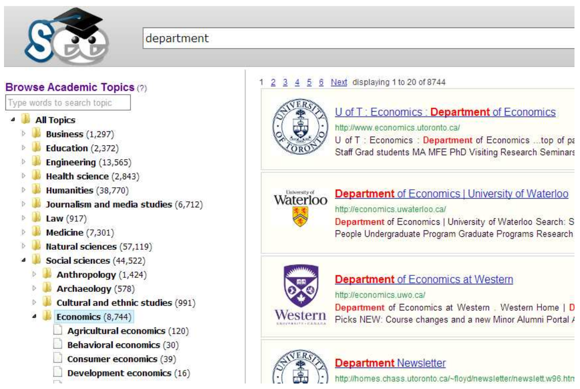
Many Canadian universities currently license Google's search with "site:" to restrict search results within their own domains. Compared with such a simple solution for university search, SEEU has four major advantages.

Firstly, SEEU can greatly facilitate research collaboration within and between universities. Faculty members and students can easily search people, departments or various research areas within a single or among multiple universities. For example, Figure 1.8(a) shows an example of searching the Department of Economics webpages, and Figure 1.8(b) shows an example of searching for professors doing research in *Computer Security* in three universities simultaneously in SEEU. Performing such search tasks in traditional general search engines without a university hierarchy could be very inconvenient. Users have to type different "site" keywords (or switch back and forth among different university search engines) to compare the results in different universities.

Secondly, in SEEU, users can find desired webpages much more easily, as they can integrate keyword search and browsing together, or use the hierarchies to filter the results. Figure 1.8(c) and Figure 1.8(d) show such examples. When searching the ambiguous keywords "active learning", researchers in different disciplines such as *Computer Science* and *Education*, can simply choose the corresponding topics to restrict the results to only within their own research areas.[8] We can see that in both cases, the top ranked results in SEEU are exclusively related to the selected category. Without the integrated topic hierarchy, such search tasks could be very difficult in flat search engines.

[7]SEEU can be visited at `http://kdd.csd.uwo.ca:88/seeu`.

[8]If users are not familiar with SEEU's topic hierarchy, they can also use the category search box above the hierarchy to locate the desired categories.



department

Browse Academic Topics (?)
Type words to search topic
All Topics
Business (1,297)
Education (2,372)
Engineering (13,565)
Health science (2,843)
Humanities (38,770)
Journalism and media studies (6,712)
Law (917)
Medicine (7,301)
Natural sciences (57,119)
Social sciences (44,522)
Anthropology (1,424)
Archaeology (578)
Cultural and ethnic studies (991)
Economics (8,744)
Agricultural economics (120)
Behavioral economics (30)
Consumer economics (39)
Development economics (16)

1 2 3 4 5 6 Next displaying 1 to 20 of 8744

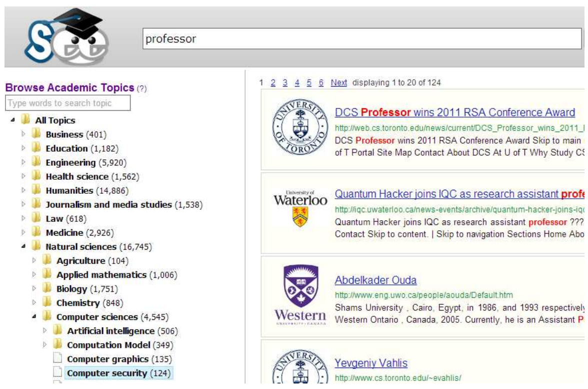
U of T : Economics : Department of Economics
http://www.economics.utoronto.ca/
U of T : Economics : Department of Economics ...top of pa
Staff Grad students MA MFE PhD Visiting Research Seminars

Department of Economics | University of Waterloo
http://economics.uwaterloo.ca/
Department of Economics | University of Waterloo Search: S
People Undergraduate Program Graduate Programs Research

Department of Economics at Western
http://economics.uwo.ca/
Department of Economics at Western . Western Home | D
Picks NEW: Course changes and a new Minor Alumni Portal

Department Newsletter
http://homes.chass.utoronto.ca/~floyd/newsletter/newslett.w96.htm

(a) Search "department" in *Economics*



professor

Browse Academic Topics (?)
Type words to search topic
All Topics
Business (401)
Education (1,182)
Engineering (5,920)
Health science (1,562)
Humanities (14,886)
Journalism and media studies (1,538)
Law (618)
Medicine (2,926)
Natural sciences (16,745)
Agriculture (104)
Applied mathematics (1,006)
Biology (1,751)
Chemistry (848)
Computer sciences (4,545)
Artificial intelligence (506)
Computation Model (349)
Computer graphics (135)
Computer security (124)

1 2 3 4 5 6 Next displaying 1 to 20 of 124

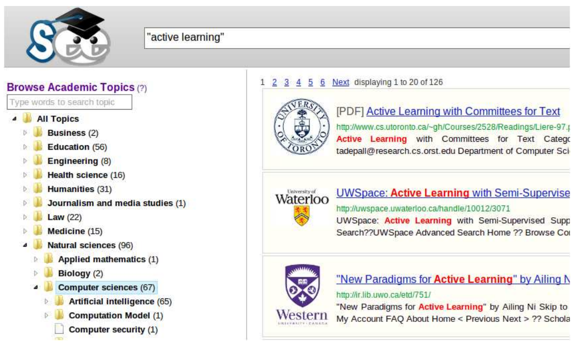
DCS Professor wins 2011 RSA Conference Award
http://web.cs.toronto.edu/news/current/DCS_Professor_wins_2011_
DCS Professor wins 2011 RSA Conference Award Skip to main
of T Portal Site Map Contact About DCS At U of T Why Study CS

Quantum Hacker joins IQC as research assistant profe
http://iqc.uwaterloo.ca/news-events/archive/quantum-hacker-joins-iqc
Quantum Hacker joins IQC as research assistant professor ???
Contact Skip to content. | Skip to navigation Sections Home Abo

Abdelkader Ouda
http://www.eng.uwo.ca/people/aouda/Default.htm
Shams University , Cairo, Egypt, in 1986, and 1993 respectively
Western Ontario , Canada, 2005. Currently, he is an Assistant P

Yevgeniy Vahlis
http://www.cs.toronto.edu/~evahlis/

(b) Search for "professor" in *Computer Security*



"active learning"

Browse Academic Topics (?)
Type words to search topic
All Topics
Business (2)
Education (56)
Engineering (8)
Health science (16)
Humanities (31)
Journalism and media studies (1)
Law (22)
Medicine (15)
Natural sciences (96)
Applied mathematics (1)
Biology (2)
Computer sciences (67)
Artificial intelligence (65)
Computation Model (1)
Computer security (1)

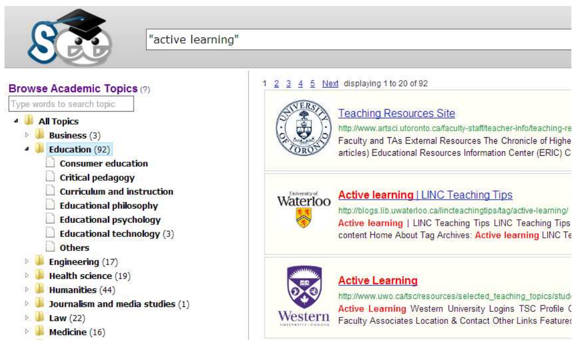
1 2 3 4 5 6 Next displaying 1 to 20 of 126

[PDF] Active Learning with Committees for Text
http://www.cs.utoronto.ca/~gh/Courses/2528/Readings/Liere-97.p
Active Learning with Committees for Text Categ
tadepall@research.cs.orst.edu Department of Computer Sci

UWSpace: Active Learning with Semi-Supervise
http://uwspace.uwaterloo.ca/handle/10012/3071
UWSpace: Active Learning with Semi-Supervised Supp
Search??UWSpace Advanced Search Home ?? Browse Co

"New Paradigms for Active Learning" by Ailing N
http://ir.lib.uwo.ca/etd/751/
"New Paradigms for Active Learning" by Ailing Ni Skip to
My Account FAQ About Home < Previous Next > ?? Schola

(c) Search "active learning" in *Comp. Science*



"active learning"

Browse Academic Topics (?)
Type words to search topic
All Topics
Business (3)
Education (92)
Consumer education
Critical pedagogy
Curriculum and instruction
Educational philosophy
Educational psychology
Educational technology (3)
Others
Engineering (17)
Health science (19)
Humanities (44)
Journalism and media studies (1)
Law (22)
Medicine (16)

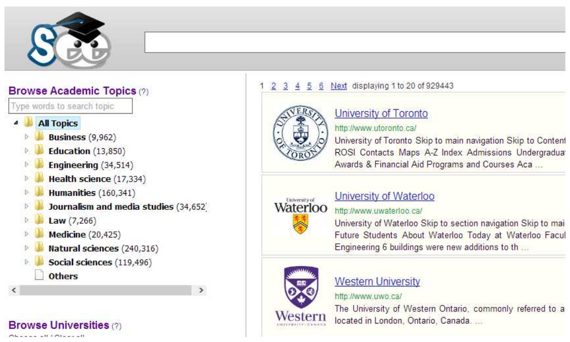
1 2 3 4 5 Next displaying 1 to 20 of 92

Teaching Resources Site
http://www.artsci.utoronto.ca/faculty-staff/teacher-info/teaching-re
Faculty and TAs External Resources The Chronicle of Highe
articles) Educational Resources Information Center (ERIC) C

Active learning | LINC Teaching Tips
http://blogs.lib.uwaterloo.ca/lincteachingtips/tag/active-learning/
Active learning | LINC Teaching Tips LINC Teaching Tips
content Home About Tag Archives: Active learning LINC Te

Active Learning
http://www.uwo.ca/tsc/resources/selected_teaching_topics/stud
Active Learning Western University Logins TSC Profile C
Faculty Associates Location & Contact Other Links Features

(d) Search "active learning" in *Education*



Browse Academic Topics (?)
Type words to search topic
All Topics
Business (9,962)
Education (13,850)
Engineering (34,514)
Health science (17,334)
Humanities (160,341)
Journalism and media studies (34,652
Law (7,266)
Medicine (20,425)
Natural sciences (240,316)
Social sciences (119,496)
Others

Browse Universities (?)

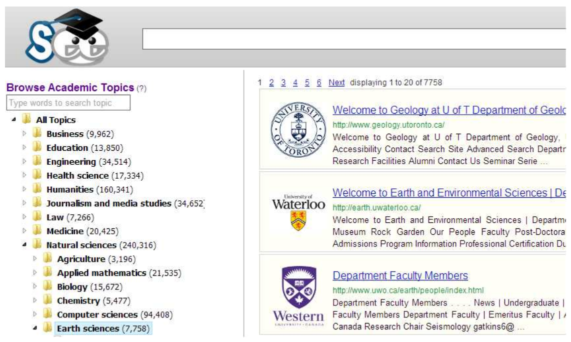
1 2 3 4 5 6 Next displaying 1 to 20 of 929443

University of Toronto
http://www.utoronto.ca/
University of Toronto Skip to main navigation Skip to Content
ROSI Contacts Maps A-Z Index Admissions Undergradua
Awards & Financial Aid Programs and Courses Aca ...

University of Waterloo
http://www.uwaterloo.ca/
University of Waterloo Skip to section navigation Skip to mai
Future Students About Waterloo Today at Waterloo Facul
Engineering 6 buildings were new additions to th ...

Western University
http://www.uwo.ca/
The University of Western Ontario, commonly referred to a
located in London, Ontario, Canada. ...

(e) Browse in flat hierarchy



Browse Academic Topics (?)
Type words to search topic
All Topics
Business (9,962)
Education (13,850)
Engineering (34,514)
Health science (17,334)
Humanities (160,341)
Journalism and media studies (34,652
Law (7,266)
Medicine (20,425)
Natural sciences (240,316)
Agriculture (3,196)
Applied mathematics (21,535)
Biology (15,672)
Chemistry (5,477)
Computer sciences (94,408)
Earth sciences (7,758)

1 2 3 4 5 6 Next displaying 1 to 20 of 7758

Welcome to Geology at U of T Department of Geolo
http://www.geology.utoronto.ca/
Welcome to Geology at U of T Department of Geology,
Accessibility Contact Search Site Advanced Search Departr
Research Facilities Alumni Contact Us Seminar Serie ...

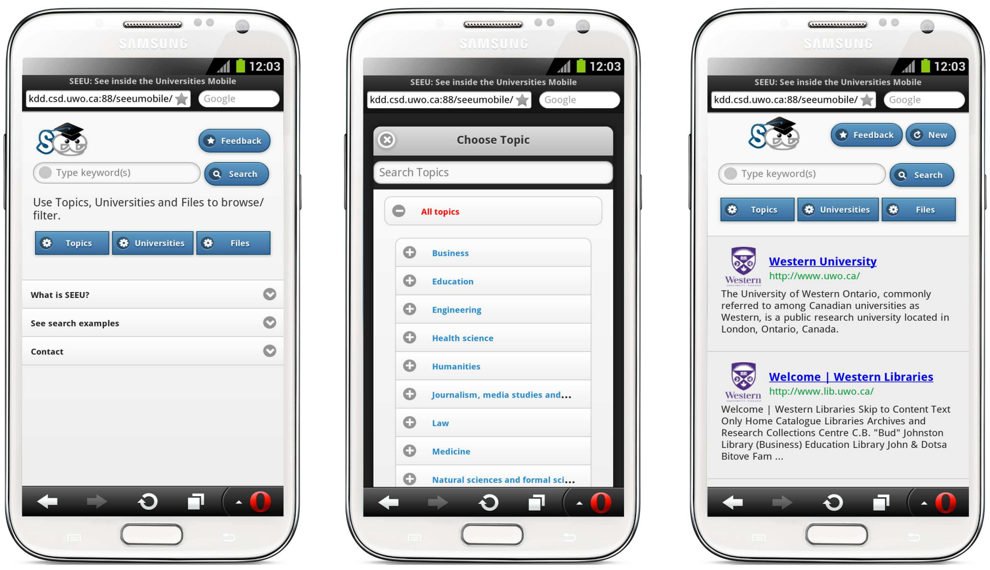
Welcome to Earth and Environmental Sciences | De
http://earth.uwaterloo.ca/
Welcome to Earth and Environmental Sciences | Departm
Museum Rock Garden Our People Faculty Post-Doctora
Admissions Program Information Professional Certification Du

Department Faculty Members
http://www.uwo.ca/earth/people/index.html
Department Faculty Members . . . . News | Undergraduate |
Faculty Members Department Faculty | Emeritus Faculty |
Canada Research Chair Seismology gatkins6@ ...

(f) Browse in *Earth Sciences*

Figure 1.8: Six usage scenarios in SEEU with Toronto, Waterloo and Western selected in the university hierarchy.

Thirdly, for some search tasks, users may not always know what keywords to use, as they may not be familiar with the domain. In this case, they can select appropriate categories (e.g., topics and universities) and browse the webpages without typing any keywords. For example, Figure 1.8(e) shows an example of browsing globally the most popular and important webpages in the 12 universities, and Figure 1.8(f) shows an example of browsing the webpages belonging to *Earth Sciences* in three universities simultaneously in SEEU. By contrast, in traditional general web search engines, no keywords often means no results.



Use Topics, Universities and Files to browse/ filter.

The University of Western Ontario, commonly referred to among Canadian universities as Western, is a public research university located in London, Ontario, Canada.

Welcome | Western Libraries Skip to Content Text Only Home Catalogue Libraries Archives and Research Collections Centre C.B. "Bud" Johnston Library (Business) Education Library John & Dotsa Bitove Fam ...

(a) Home page (b) Topic hierarchy dialog (c) Result page

Figure 1.9: The home page, the topic hierarchy dialog and the result page in SEEU Mobile. The university dialog and the file dialog are similar to the topic dialog.

The fourth major advantage is that SEEU is particularly suitable for small-screen devices, as browsing is much easier than typing keyword(s) on small keyboards [64, 52]. The user interface of SEEU can automatically adapt to devices with small screens without losing UI consistency (see Figure 1.9 for the mobile user interface of SEEU).[9] Specifically, due to the limited screen size, the three hierarchies of SEEU are replaced by three buttons on the home page. When users click one of the buttons (say Topics), SEEU Mobile opens a popup dialog (of full screen size) (see Figure 1.9(b)) that only shows the current selected category and its child subcategories in the hierarchy. Users can click a desired subcategory to browse or filter the search results. To the best of our knowledge, none of the search engines currently adopted by Canadian universities (on their home pages) provide mobile-specific user interfaces.

### 1.3.2 Challenges of Building SEEU

The novel and key idea of SEEU is to leverage multiple relevant hierarchies to search university webpages. Among the three hierarchies (i.e., topics, universities and media types) in SEEU, the most powerful yet difficult one is the topic hierarchy. There exist three major challenges to integrate the topic hierarchy into SEEU.

1. How to define a commonly accepted academic topic hierarchy for university webpages?

2. How to effectively categorize millions of university webpages into the hierarchy?

3. How to effectively rank webpages in each category of the hierarchy?

[9]Both PC and mobile versions of SEEU have the same entry link as `http://kdd.csd.uwo.ca:88/seeu`. When users visit SEEU on mobile devices, they will be automatically redirected to the mobile version of SEEU.

The first challenge is to define a commonly accepted academic topic hierarchy for university webpages. A reasonable and user-friendly hierarchy of topics is very important for university search. It can greatly benefit the search experience of the users. To build such a hierarchy for universities, we need to consider at least two important criteria. Firstly, the topic hierarchy must be academe related. It is desirable to present a hierarchy with a broad coverage of common academic disciplines so that the majority of university users will feel it is familiar and convenient in finding the desired topics. The second criterion is that the hierarchy should be friendly for users to browse. Usually, the tree-structured hierarchies are intuitive and acceptable in most web-based applications. However, a complicated hierarchical structure (with deep categories) may be too complex for normal users. Thus, for convenience consideration, the hierarchy should not be too deep.

The second challenge is to design an effective method to classify millions of webpages into the predefined hierarchy. As we discussed in Section 1.2, manual classification of webpages is hopeless due to the large number of webpages and the exponential growth rate. Traditional text classification methods can only classify documents into a few classes [57, 93, 113, 120]. However, in SEEU, the topic hierarchy may contain a large number of categories. Designing an effective classification algorithm that can scale to millions of webpages in a large hierarchy is crucial. In addition, the algorithm should also consider the hierarchical relationship between different categories. It is useless to predict contradicted categories (e.g., a webpage predicted to "Machine Learning" but not "Computer Science").

The third challenge is to effectively rank webpages in each category of the hierarchy. Traditional keywords-based search engines usually only rely on keywords matching and page importance metrics (e.g., PageRank [85]) to rank webpages. However, in SEEU, when a user searches webpages in a category of the hierarchy, we also need to consider the category relevance of webpages. Usually, given a category of the hierarchy, not all webpages are equally relevant to it. For example, consider a case when we search the keywords "active learning" inside the category "Computer Science". There may exist some webpages that simply mention "active learning" in the text. Their relevance to "Computer Science" may not be as high as the theoretical research works on "active learning". Thus, SEEU should also take category relevance into consideration.

## 1.4 Contributions of the Thesis

Effective information organization and retrieval in universities is important. Current search engine approaches have limitations in dealing with the challenges of searching university webpages. Integrating topic hierarchies and keyword search is known to improve the users' search experience in E-commerce web sites. A major challenge in doing so for university webpages is to define a commonly accepted academic topic hierarchy, and effectively classify and rank a massive number of university webpages into the hierarchy.

In this thesis, we propose a novel search engine approach for universities, called SEEU[10] (**S**earch **E**ngine with hi**E**rarchies for **U**niversities), for Canadian universities to facilitate research collaboration and help people find desired university webpages more easily. The main

[10]SEEU can be visited at `http://kdd.csd.uwo.ca:88/seeu`.

contribution of this thesis are listed as follows.

1. An effective hierarchical webpage classification system. It contains a webpage feature extraction tool based on Hadoop MapReduce, and a parallel hierarchical SVM (Support Vector Machine) classifiers based on OpenMPI. Part of this work was published in the *Proceeding of the 22nd Internation Joint Conference on Artificial Intelligence* (IJCAI 2011) [67].

2. ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), a new ranking framework for search engines with hierarchies. It integrates hierarchical classification probabilities, keywords relevance and document related metrics into a learning to rank [45, 62, 18] framework. This work was submitted to the *IEEE Transactions on Knowledge and Data Engineering* (IEEE TKDE) [75].

3. A novel active learning framework for hierarchical text classification. It leverages the top-down tree structure to coordinate classification system and data labeling source on limited labeled datasets. This work was published in the *Proceeding of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD 2013) [76]. An earlier work was published in the *Proceeding of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD 2012) [74].

4. A novel application of the proposed classification and ranking methods for mining academic topics in universities.

5. A prototype of SEEU search engine and two usability studies of SEEU in our university. This work was included in our IEEE TKDE paper [75].

The relation of all contributions can be visualized in Figure 1.10.

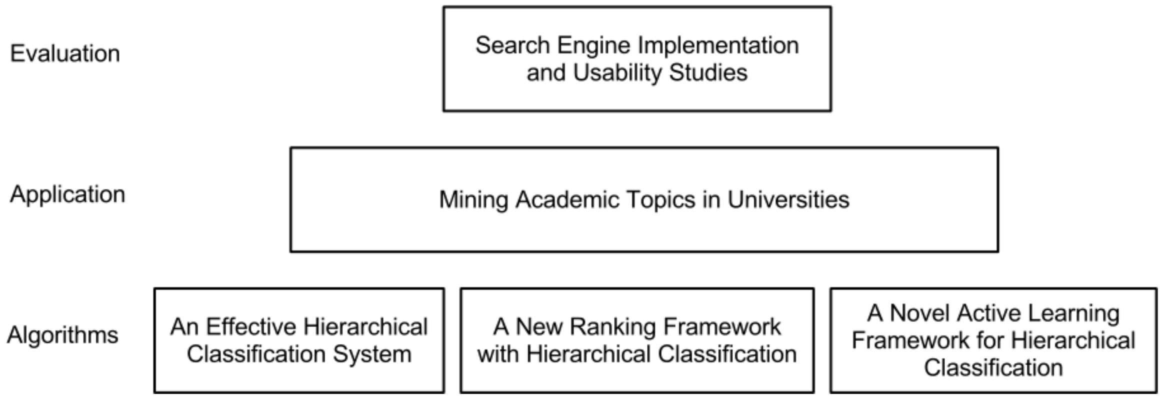

Evaluation
Search Engine Implementation and Usability Studies
Application
Mining Academic Topics in Universities
Algorithms
An Effective Hierarchical Classification System
A New Ranking Framework with Hierarchical Classification
A Novel Active Learning Framework for Hierarchical Classification

Figure 1.10: The relation of all contributions.

## 1.5 Thesis Outline

This thesis is organized as follows. In Chapter 2, we review the related works in the field of text classification, hierarchical text classification, learning to rank and topic modeling. In Chapter

3, we describe an implementation of an effective hierarchical webpage classification system. In Chapter 4, we propose the ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), for search engines with hierarchies. In Chapter 5, we propose a novel active learning framework for hierarchical text classification. In Chapter 6, we develop a novel application based on the proposed classification and ranking methods to mine academic topics in universities. In Chapter 7, we present the system implementation and conduct two usability studies to evaluate SEEU. We end this thesis with conclusions and a discussion of future works in Chapter 8.

Part of the work presented in this thesis is in collaboration with Dr. Charles Ling, Da Kuang and Dr. Huaimin Wang.

# Chapter 2

# Related Work

In this chapter, we review previous work related to our work. Firstly, as we study document classification and retrieval in this thesis, we review the state-of-the-art text classification methods in Section 2.1. Secondly, we review previous work in hierarchical classification related to the hierarchical browsing and search in SEEU in Section 2.2. Thirdly, in Section 2.3, we review previous work in learning to rank which is related to the ranking of documents in each category of the topic hierarchy in SEEU. Finally, in Section 2.4, we review topic modeling techniques, which are related to mining academic topics in universities in SEEU.

It should be noted that we mainly review *general* topics in each research area in this chapter. Specific work which is directly compared with ours, will be reviewed in greater detail in the later chapters.

## 2.1 Text Classification

In SEEU, we are facing the problem of webpage classification. Due to the explosive growth of documents (webpage) on the web, manual classification is hopeless. Most text classification applications on the web (e.g., spam filtering [2, 98], news article categorization [71, 79] and sentiment classification [86, 65]) usually adopt automatic text classification by supervised text classification methods.

In supervised text classification, we train a classification model from a set of training data with labels provided. Usually, the labels are predefined in a category (class) space. Thus, the task of supervised text classification is to label documents with one or more predefined classes. If the category space has only two classes, the classification task is called *binary classification*. For classification problems with more than two classes, if a document can only be assigned to one class, the classification task is called *multi-class classification* [57, 93]; if a document can be assigned to multiple classes, it is called *multi-label classification* [113, 120, 82, 124].

We give a formal definition of supervised text classification in the framework of *binary classification*. For *multi-class classification* and *multi-label classification*, we can decompose the learning task into several *binary classification* subtasks by training a binary classifier for each class (i.e., one-vs-rest strategy) [93, 113].

Consider a dataset from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the example set and $\mathcal{Y}$ is the label set $\{1, -1\}$. We usually call the example with label 1 a positive example and the example with label

-1 a negative example. We assume that a training set $D_{train} = \{(x_i, y_i)\} \subset \mathcal{X} \times \mathcal{Y}$ is given. Each $(x_i, y_i)$ in $D_{train}$ denotes a training example $x_i$ and the associated label $y_i$. The classification problem is to learn a decision function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from the training set $D_{train}$ with good classification performance on a separate testing set $D_{test} = \{(x_i, y_i)\} \subset \mathcal{X} \times \mathcal{Y}$.

In the next subsections, we will firstly review the three most popular text classification algorithms that learn the decision function $f$. They are *Naive Bayes*, *K-nearest Neighbor* and *Support Vector Machine*. After that, we review the methods that evaluate text classification algorithms on the testing set $D_{test}$.

### 2.1.1 Naive Bayes

In *Naive Bayes* [83], given an example $x$, the decision output $f(x)$ is learned by maximizing the posterior probability $P(y|x)$:

$$f(x) = \arg\max_{y \in \mathcal{Y}} P(y|x) \tag{2.1}$$

By applying the Bayes rule, this equation can be rewritten as:

$$f(x) = \arg\max_{y \in \mathcal{Y}} P(y|x) = \arg\max_{y \in \mathcal{Y}} \frac{P(y)P(x|y)}{P(x)} \tag{2.2}$$

As $P(x)$ is constant, we can simply remove it and yield the following simpler form

$$f(x) = \arg\max_{y \in \mathcal{Y}} P(y)P(x|y) \tag{2.3}$$

Therefore, to learn the classification model of *Naive Bayes*, we need to learn the conditional probability $P(x|y)$ and the prior probability $P(y)$.

It turns out that calculating $P(x|y)$ is difficult as we do not know the distribution of $x$. To deal with this issue, *Naive Bayes* makes a *naive* assumption that each document $x$ can be represented by a bag of words $\{w_1, w_2, \ldots, w_n\}$ from a vocabulary $V$ and the words in the document are independent of each other [70]. Thus, the decision function $f(x)$ can be rewritten as

$$f(x) = \arg\max_{y \in \mathcal{Y}} P(y)P(x|y) = \arg\max_{y \in \mathcal{Y}} P(y) \prod_{i=1}^{n} P(w_i|y) \tag{2.4}$$

Finally, we can compute the decision output $f(x)$ as

$$f(x) = \begin{cases} 1, & P(y=1)\prod_{i=1}^{n} P(w_i|y=1) > P(y=-1)\prod_{i=1}^{n} P(w_i|y=-1) \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

The posterior probability $P(y|x)$ for each class $y$ can be obtained as

$$P(y=i|x) = \frac{P(y=i)\prod_{i=1}^{n} P(w_i|y=i)}{\sum_{i \in \{1,-1\}} P(y=i)\prod_{i=1}^{n} P(w_i|y=i)} \tag{2.6}$$

To train a *Naive Bayes* model, we learn the parameters $P(y)$ and $P(w_i|y)$ by the maximal likelihood principle. Specifically, we can calculate $P(y)$ based on the empirical frequency of $y$

in the training set as $N(y)/m$, and calculate the posterior probability $P(w_i|y)$ of each word $w_i$ as $N(w_i, y)/\sum_{w_i \in V} N(w_i, y)$, where $N(y)$ is the total number of examples with label $y$ and $N(w_i, y)$ is the total occurrences of word $w_i$ in the examples with label $y$. These values can be easily calculated by counting the frequency in the training set.

It should be noted that the conditional probability $P(w_i|y)$ is calculated only from the training set. When we apply *Naive Bayes* to new documents, it is very likely that we will see unknown words which will result in a zero probability (i.e.,$N(w_i, y)/\sum_{w_i \in V} N(w_i, y) = 0$). This is undesirable. A common way to solve this problem is to use *laplace smoothing*:

$$P(w_i|y) = \frac{N(w_i, y) + k}{\sum_{w_i \in V} N(w_i, y) + k * |V|}, \tag{2.7}$$

where k is the laplace parameter and $|V|$ is the vocabulary size. Usually, we set the parameter $k$ to 1. That means we add one pseudo example to each word in the vocabulary.

There are two advantages of *Naive Bayes*. Firstly, it is very efficient to train the classification model as we only need to scan the training set once to count word frequency. With the recent advances in Big-data platform such as MapReduce [32], we can efficiently train *Naive Bayes* on hundreds of thousands of documents. Secondly, predicting new documents is also very efficient. As the parameters are already calculated, a simple multiplication of the prior probability and the conditional probabilities of each word can yield the prediction.

However, a major disadvantage of *Naive Bayes* is the unrealistic independence assumption. *Naive Bayes* assumes that the words of a document are independent. In real-world application, the words are often correlated, such as “machine learning” and “data mining”. Simply discarding such correlation may cause the posterior probability $P(y|x)$ to be not well calibrated [5].

### 2.1.2 K Nearest Neighbor

*K Nearest Neighbor* (*KNN*) [48] is an instance-based, or lazy-learning classification algorithm. Unlike *Naive Bayes*, *KNN* does not need a training stage to build the classification model. The main computational cost of *KNN* is in the prediction stage (i.e., classifying new examples).

The general idea of *KNN* is very simple. Given an example $x$, *KNN* firstly finds k nearest neighbors for this example and then classifies as to the most common class by *majority vote* among the k neighbors. More formally, the decision function $f(x)$ of *KNN* can be defined as

$$f(x;k) = \arg\max_{y \in \mathcal{Y}} |\{x_i \in \mathcal{N}(x;k) | y_i = y\}|, \tag{2.8}$$

where $k$ is the parameter of *KNN* and $\mathcal{N}(x;k)$ is the $k$ nearest neighbors of example $x$.

The standard *majority vote* method may not work well in imbalanced datasets where examples from the major class will dominate the k neighbors. To deal with this issue, we can weigh the vote of each neighbor by its distance to the testing example. The intuitive idea is that closer examples should have higher weights than more distant examples. The weights can be computed as the inverse of any distance metrics, such as the *Euclidean distance*.

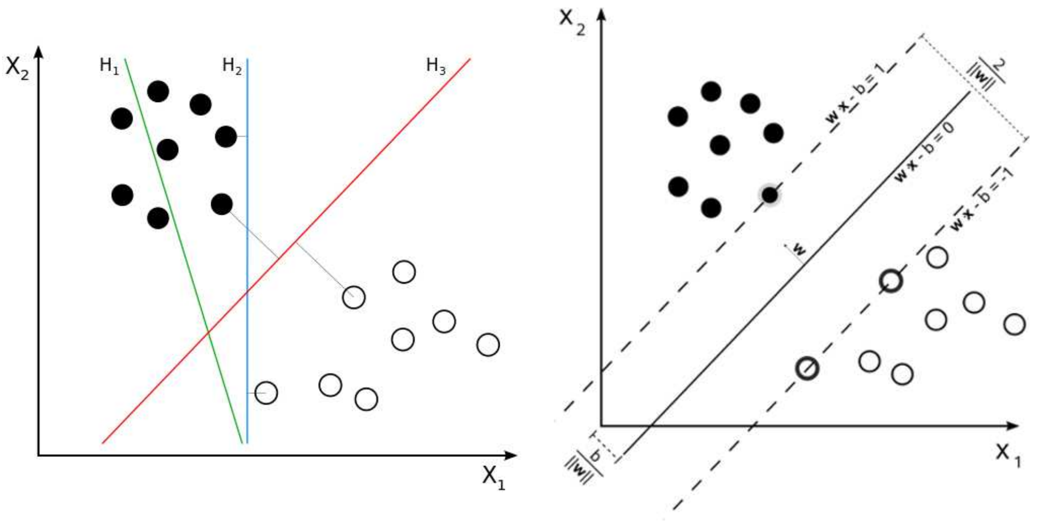
Regarding the advantages of *KNN*, we can see that it is very simple to implement. For high-level programming languages, such as Matlab, the implementation of *KNN* only takes several lines of code. In addition, the classifier can be updated online as there is no cost in training.

However *KNN* has several weaknesses in text classification. Firstly, *KNN* is very sensitive to the noisy and irrelevant features. The prediction accuracy of *KNN* can quickly degrade when the total number of features grows. Secondly, as a lazy-learning algorithm, *KNN* requires storing the training examples. This could be of great cost and thus makes *KNN* difficult to scale to large datasets.

### 2.1.3 Support Vector Machine

*Support Vector Machines* (*SVMs*) are currently the state-of-the-art text classification algorithms [60, 61, 20, 121]. SEEU uses *SVM* as the base classifier for document classification. We will give a detailed description of *SVM*.

In *SVM* [26], each example $x$ in the training set is represented as a point in a high dimensional feature space. The basic idea of *SVM* is to find a hyperplane that separates the positive examples and the negative examples in the training set with the largest margin (i.e., distance to the boundary of each class). For example, in Figure 2.1a, we can see three hyperplanes $H_1$, $H_2$ and $H_3$. $H_2$ and $H_3$ can separate the two classes. However, only $H_3$ is the *maximal margin* hyperplane as demonstrated in Figure 2.1b.



(a) SVM separating hyperplanes (b) SVM maximum-margin hyperplane

Figure 2.1: Support Vector Machine.

We describe the formation of the *maximal margin* principle. Formally, given an example $x$, the decision hyperplane (function) $f$ of a *SVM* is defined as the equation $f(x; w) = w \cdot x - b = 0$ where $w$ and $b$ are the parameters. Let us assume that the dataset is linearly separable. This is usually true as the high dimensionality of text features usually results in the dataset being linearly separable [60]. We need to find two hyperplanes parallel to $f$ such that no training examples fall between them (perfect classification) and their distance to $f$ is maximal. Mathematically, the two hyperplanes can be defined as $w \cdot x - b = 1$ and $w \cdot x - b = -1$ respectively. Thus, the so-called margin (geometric distance between the two hyperplans) of *SVM* can be calculated as $2/||w||$.

Therefore, training a *SVM* means to find the best $w$ and $b$ to minimize $||w||$. To avoid the difficulty of dealing with the square root in $||w||$, we usually change it to $\frac{1}{2}||w||^2$. In addition, to ensure that no training examples lie in the margin, we need to add constrains for each training example $x_i$. That is, $y_i(w \cdot x_i - b) \geq 1$. Thus, training an *SVM* can be formalized as an optimization problem:

$$\begin{gathered} \min_{w,b} \frac{1}{2} \parallel w \parallel^2 \\ subject\ to\ y_i(w \cdot x_i - b) \geq 1,\ \forall i\ (1 \leq i \leq m) \end{gathered} \tag{2.9}$$

It may be argued that the assumption of perfect linear separability is unrealistic in a real-world application. This is usually due to the mislabeled examples or noise in the training set. Cortes [26] proposes the soft margin approach by introducing the slack variables $\xi$ which measure the degree of the misclassification of training examples. The optimization problem becomes

$$\begin{gathered} \min_{w,b,\zeta} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} \xi_i \\ subject\ to,\ y_i(w \cdot x_i - b) \geq 1 - \xi_i,\ \forall i\ (1 \leq i \leq m) \end{gathered} \tag{2.10}$$

where $C$ is the parameter for tradeoff between the maximal margin and the minimal classification error.

Many approaches have been proposed for solving this optimization problem, such as Sequential Minimal Optimization [87], Stochastic Gradient Descent [125, 102] and Dual Coordinate Descent [56]. The typical software packages for training *SVM* include SVMLight [61], LibSVM [23] and LiblinearSVM [41].

We list the advantages of using *SVM* as the text classification algorithm.

1. **Good generalization capability**. *SVM* has good generalization capability as *SVM* tries to maximize the margin between the positive and the negative examples. Moreover, with the soft margin [26] introduced in the optimization equation, *SVM* is even robust against the noisy data in the training set.

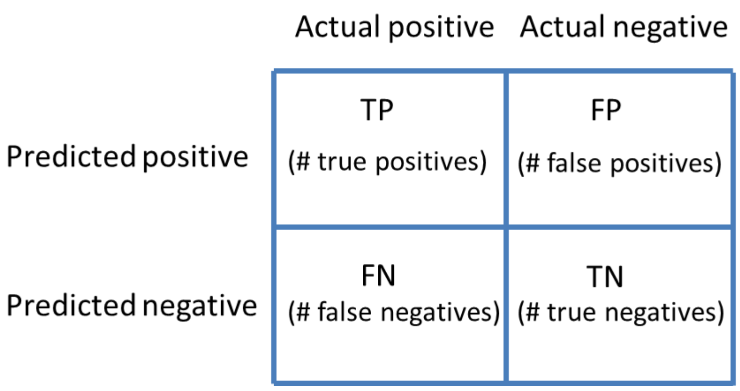
2. **Effective text classification**. Usually, the word vocabulary for text data is very large. The high dimensionality of text datasets often leads to the training data being linearly separable [61, 41]. Thus, *SVM* with linear kernel (e.g., LiblinearSVM) is often very suitable for text classification task.

Due to the advantages mentioned above, *SVM* is a good choice for our task. Furthermore, previous work [60, 61, 20, 121] empirically verifies that *SVM* is superior to other classification algorithms such as *Naive Bayes* and *KNN*, in terms of text classification performance. Thus, we choose *SVM*, specifically linear *SVM*, as the base classification algorithm to classify webpages into search engines (see Chapter 3).

### 2.1.4 Evaluation Measures

In this subsection, we review methods to evaluate the classification performance of classifiers on the testing set. We firstly introduce a tool, called a confusion matrix [107], for performance analysis. The confusion matrix is a table with two rows and two columns (see Figure 2.2). Each cell in the table reports the number of specific prediction judgements, including true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN).

Each of the four statistics is computed based on the classifier prediction and the actual true classes. For example, given a classifier $f$ and a testing dataset $D_{test}$, TP can be computed as $\sum_{i=1}^{|D_{test}|} \mathbb{1}(f(x_i) = 1 \text{ and } y_i = 1)$ where $\mathbb{1}$ is the indicator function with value 0 and 1 defined on the logic expression "$f(x_i) = 1$ and $y_i = 1$".



| | Actual positive | Actual negative |
|---|---|---|
| Predicted positive | TP (# true positives) | FP (# false positives) |
| Predicted negative | FN (# false negatives) | TN (# true negatives) |

Figure 2.2: Confusion matrix.

The simplest performance measure is the accuracy, which is defined as the proportion of correctly classified examples in the testing set:

$$M_{accuracy}(f) = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.11}$$

Although the definition of accuracy is very intuitive, it is rarely used in real-world text classification due to the class imbalance problem [58]. For example, given a spam classification dataset with only ten positive examples (i.e., spam mails) and 90 negative examples (e.g., normal mails), it is trivial to achieve 90% accuracy by predicting all examples as negative (i.e., normal mails). However, such a *highly accurate* algorithm is useless for spam classification.

To make a realistic evaluation of classification performance on imbalanced datasets, people have developed several more effective measures. The most popular measures are precision and recall.

$$M_{precision}(f) = \frac{TP}{TP + FP} \tag{2.12}$$

$$M_{recall}(f) = \frac{TP}{TP + FN} \tag{2.13}$$

We can see that precision is actually the ratio of correctly predicted positive examples over all positive predictions; and recall is actually the ratio of correctly predicted positive examples over all actual positive examples.

We use an example to show the difference among the three measures. Consider again the spam classification dataset with ten positive examples and 90 negative examples. We assume that a classifier $f$ makes a confusion matrix as shown in Table 2.1.

Table 2.1: An example of confusion matrix.

| | Actual positive | Actual negative |
|---|---|---|
| Predicted positive | TP=1 | FP=3 |
| Predicted negative | FN=9 | TN=87 |

Then, we have $M_{accuracy} = (1+87)/(1+87+3+9) = 0.88$, $M_{precision} = 1/(1+3) = 0.25$ and $M_{recall} = 1/(1+9) = 0.1$. The high accuracy is actually very misleading while the performance measures based on precision and recall are more realistic. Specifically, for the reported spam mails, only 25% prediction is correct based on precision. Based on recall, we can find that this classifier can only detect 10% spam mails. The results based on precision and recall make us reject this spam classifier for real-world deployment.

Usually, it is more convenient to judge classification performance by a single measure. To make a trade-off between precision and recall, we often use the harmonic mean of precision and recall, called F1-score [121], in text classification:

$$M_{f1-score}(f) = \frac{2}{\frac{1}{M_{precision}(f)} + \frac{1}{M_{recall}(f)}} = \frac{2 \times M_{precision}(f) \times M_{recall}(f)}{M_{precision}(f) + M_{recall}(f)} \tag{2.14}$$

Figure 2.3 plots the 3D value curve of a F1-score based on precision and recall. We can see that to achieve a high F1-score, classifiers must have both high precision and high recall. Optimizing classifiers on a single measure regardless of the other measure, will not improve F1-score much. For example, for the spam classification dataset we discussed before, if a classifier predicts all examples as positive, it can achieve 100% recall but the precision will be only 10%. This results in the F1-score as low as 0.18.

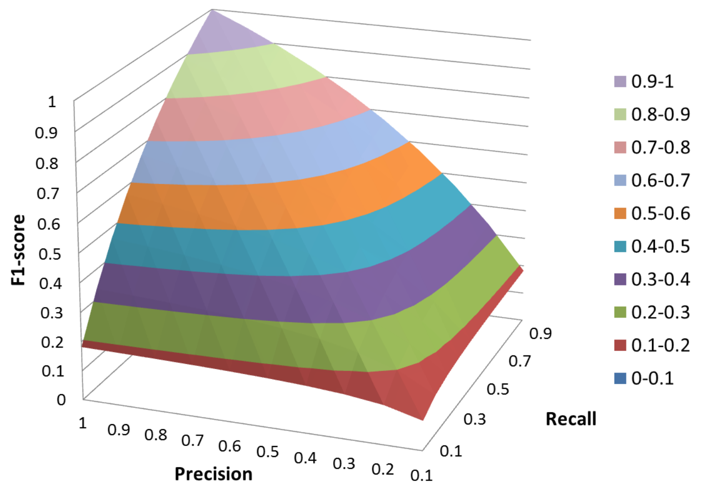


Figure 2.3: The value curve of F1-score based on precision and recall.

To conclude, based on the discussion above, we mainly use the F1-score to evaluate classification performance. Precision and recall will also be used to explain the results of the F1-score.

## 2.2 Hierarchical Text Classification

We choose *Support Vector Machine* (*SVM*) as the base classification algorithm in SEEU. However, *SVM* is a binary classifier which can only categorize two classes. In hierarchical text classification, the taxonomy (hierarchy) usually contains a large number of categories. How can we adopt *SVM* to categorize text documents into very large hierarchies? In this section, we review the previous work in hierarchical text classification. We will present our hierarchical classification system in Chapter 3.

According to Silla [106], the approaches in dealing with hierarchical classification can be generally categorized into the flat classification approach, the top-down approach, and the global classifier (or Big-bang) approach. In the following subsections, we will review the three approaches briefly.

### 2.2.1 Flat Classification Approach

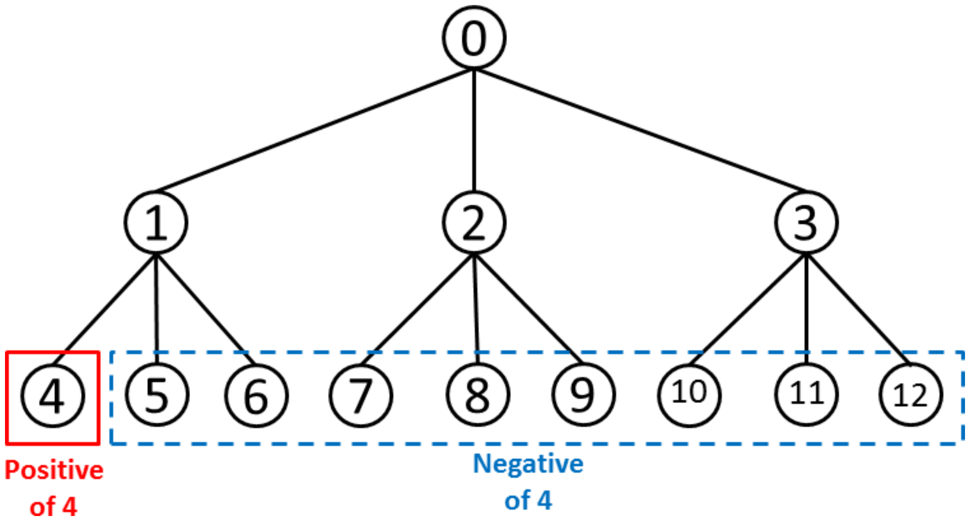


Figure 2.4: An example of the flat classification approach.

The flat classification approach [4, 49] is the simplest method, which ignores the hierarchical structure of the categories. This approach usually adopts the one-vs-rest strategy to decompose the hierarchical classification task into multiple binary classification tasks on the leaf categories. Specifically, the flat approach builds a binary classifier at each leaf category to distinguish all the other leaf categories. For example, given a hierarchy as shown in Figure 2.4, the flat approach builds a binary classifier at the 4th category by using the examples belonging to 4 as positive training examples and the examples at the other leaf categories (i.e., from the 5th to the 12th category) as negative training examples. During the prediction phase, if an example is classified into the 4th category, based on the "IS-A"[1] relation in the hierarchy, the example will also be categorized to the 1st category.

We can see that as a simple learning method, the flat approach is very easy to implement by adopting existing binary classification algorithms. However, the flat approach has several

[1]"IS-A" is a relationship where a category A is a subcategory of another category B.

weaknesses. Firstly, the positive and negative class of the training set for each classifier are usually very unbalanced. For example, in Figure 2.4, the negative class of the 4th category is eight times larger than the positive class. This imbalance problem will make it difficult to learn a good classification model. Secondly, as each classifier in the flat approach is trained on the entire dataset, the computational effort can be very high for to a large hierarchy with hundreds of leaf categories. Thirdly, the flat approach assumes that all the examples must be predicted into the leaf categories while many real-world applications are actually non-mandatory leaf-category classification tasks [106]. That is, the example does not need to be classified at the leaf categories. For example, given a hierarchy containing a path "Science"→"Computer Science"→"Artificial Intelligence", the home page of the Computer Science department should only be categorized as "Science"→"Computer Science", not "Artificial Intelligence".

### 2.2.2 Top-down Classification Approach

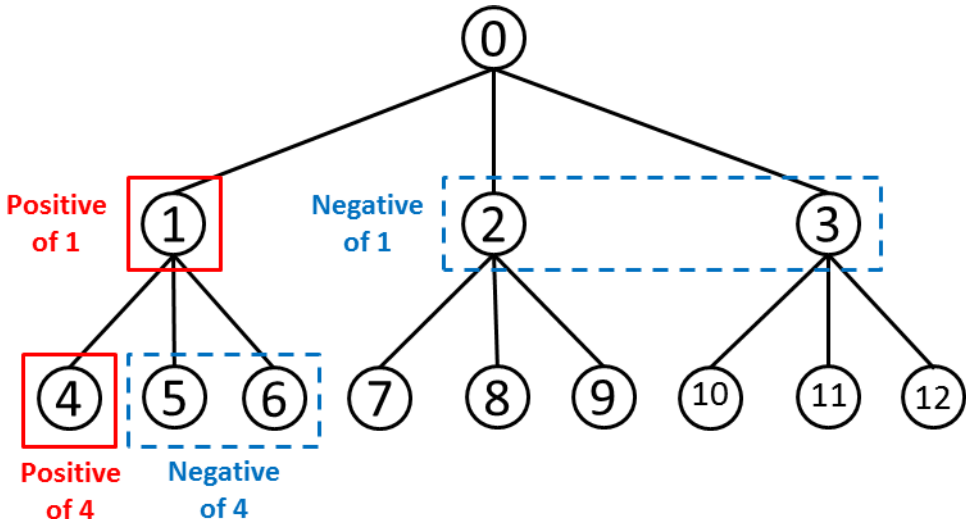


Figure 2.5: An example of the top-down classification approach.

Unlike the flat approach, which only builds binary classifiers at the leaf categories, the top-down approach [37, 110, 109, 81] builds a binary classifier on each category of the hierarchy. Thus, by setting a proper prediction threshold on each category, the top-down approach can classify examples into internal categories [110, 22]. It may be argued that the top-down approach is not as efficient as so many classifiers are trained on the hierarchy. To tackle this problem, the top-down approach builds the training set for each category *locally* by only using the examples belonging to its parent category. For example, in Figure 2.5, for the 4th category, its training set consists of the examples belonging to the 4th category (as positive examples) and the examples belonging to the 5th and the 6th category (as negative examples). Compared with Figure 2.4, we can see that a large portion of the negative examples from the 7th to the 12th category are excluded.

There are two benefits of this strategy. First, the training cost of the top-down approach is much smaller than the flat approach. At each category of the hierarchy, as the negative examples are only selected from the examples belonging to the parent category, the size of the training set is exponentially smaller than the flat approach. Second, the top-down approach will not suffer from the serious class imbalance problem. As a large number of negative examples are excluded from the training set, the class distribution of training sets in the top-down approach is more balanced than the flat approach. This will eventually lead to better classification performance.

In the prediction phase, the top-down approach predicts examples from the top-level to the bottom level. Specifically, given a testing example, the classifiers at the top level first predict it as positive or negative based on prediction thresholds.[2] After that, at each category, if the example is predicted as positive, the top-down approach will *recursively* push it down to the lower-level child categories for further prediction until the leaf categories are reached.

There are also several weaknesses of the top-down approach. First, the classification performance at the deep categories may not perform well due to the small size of positive training examples. With very limited training examples at deep categories, it may be difficult to achieve accurate classification performance. Second, the top-down prediction algorithm may be suboptimal. The top-down approach uses thresholds to filter examples from the top level to the bottom levels. Sometimes, a high threshold may cause examples blocked at the top level, while a low threshold will introduce wrong prediction into the lower levels. This is usually called the false negative and the false positive tradeoff in the top-down approach. Some methods have been proposed to tackle this problem, such as *Blocking* [109] and *Refined Experts* [7]. However, both approaches require training another hierarchical classifier to refine the results. This is not practical for very large hierarchies.

### 2.2.3 Global Classification Approach

The global classifier approach basically constructs only one classification model for the entire hierarchy. This is different from the flat approach and the top-down approaches, where many binary classifiers need to be built. Some hierarchical classification algorithms can be categorized as global classifier approaches, such as the HMC decision tree [114], the hierarchical kernel classifier [95] and the deep classifier [119]. In the training phase, these methods often rely on various special algorithms to cope with the hierarchical relations between categories. For example, the HMC decision tree uses Predictive Clustering Trees (PCT) [12] to learn optimal attribute-value tests to partition training set into hierarchical clusters. These attribute-value tests can be used to categorize examples into clusters where the major category in a cluster is used as a prediction result. The deep classifier proposes a search based method to reduce a very large and deep hierarchy into a small and shallow hierarchy, and build a multi-class Naive Bayes prediction model. The hierarchical relations between different categories are further leveraged to increase the positive examples of deep categories.

All the global approaches can directly output a subset of the hierarchy as prediction in the testing phase. Thus, the prediction time of the global approach is smaller than the flat and the top-down approaches. However, a major drawback of the global approach is that it may be difficult to model complicated relations among categories in a single classification model. In addition, the global approach also lacks the modularity of the top-down approach. When we

[2]If the prediction probability is larger (smaller) than the prediction threshold, the example will be classified as positive (negative).

add a new category into the hierarchy, we only need to add a new binary classifier for the top-down approach. However, for the global approach, we may have to retrain the entire classifier again on the new hierarchy.

To summarize, in this section, we have reviewed three hierarchical classification approaches, namely the flat approach, the top-down approach and the global approach. Previous work [31, 81, 33] has shown that the top-down approach performs better than the flat classification approach. Moreover, the top-down approach is reported to work effectively on very large hierarchies [81]. Therefore, we choose the top-down approach for hierarchical text classification in SEEU (see Chapter 3).

## 2.3 Learning to Rank

Effectively ranking documents in each category of the hierarchy is important in search engines with hierarchies. In this section, we review previous work on learning to rank. We present our ranking system in Chapter 4.

In the information retrieval literature, learning to rank means using machine learning methods to rank documents in search engines according to their relevance to the queries. Formally, consider a training set $D_{train} = \{(q_i, D_i, \preccurlyeq_i) | 1 \leq i \leq m\}$ where each training example $(q_i, D_i, \preccurlyeq_i)$ consists of a query $q_i$, a document list $D_i = \{d_1, d_2, d_3, \ldots, d_m\}$, and an optimal ranking $\preccurlyeq_i$ (i.e., $d_a \preccurlyeq_i d_b$ if and only if $d_a$ is more relevant to $q_i$ than $d_b$). We want to learn a function $f$ on $D_{train}$ that can output a good ranking for a pair of a query and a document list in the testing set $D_{test}$.

Many learning algorithms have been proposed for the ranking problem in recent years. Based on the input-output representation and the loss optimization methods, these approaches can be categorized into three groups, namely the pointwise approach, the pairwise approach and the listwise approach.

### 2.3.1 Pointwise approach

Given a query, the pointwise approach assumes that each document in a training example can be assigned a relevance score. Thus, the ranking problem can be approximately tackled by pointwise optimization methods (i.e., optimizing on individual documents), such as *regression* [45, 27], *multi-class classification* [73] and *ordinal regression* [28, 25]. We tabulate the difference of the three methods in Table 2.2.

Table 2.2: Classification of pointwise approach.

| Method | Input | Output | Loss Function |
|---|---|---|---|
| Regression | Single document | Real values | Regression loss |
| Multi-class classification | Single document | Categories | Classification loss |
| Ordinal regression | Single document | Ordinal categories | Ordinal regression loss |

The *regression* method is the simplest pointwise ranking approach. It learns a regression function by minimizing the regression loss (e.g., mean squared error) [27]. Given a query, after we use a regression function to output regression (numerical scores) for all the indexed documents, it is straightforward to rank documents by these numerical scores.

The *multi-class classification* method assumes that the relevance of documents can be categorized into several categories, such as *irrelevant*, *relevant* and *highly relevant*. Thus, the ranking problem can be cast as a multi-class classification problem. This is motivated by the fact that perfect classification leads to perfect ranking [73].

The *ordinal regression* method assumes that the relevance scores of documents are ordinal categories (e.g., *highly relevant* $\preccurlyeq$ *relevant* $\preccurlyeq$ *irrelevant*). Thus, the *ordinal regression* method can be seen as a hybrid of *regression* and *multi-class classification*. For example, given $n$ ordinal categories, Crammer [28] firstly learns a regression function on the training set. After that, it learns $n$ intervals (e.g., $a_i \leq f(x) \leq b_i$, $(1 \leq i \leq n)$) each of which represents an ordinal category. Based on this, a large margin principle is further proposed to maximize the margin between adjacent ordinal categories (e.g., $\sum_{i=2}^{n}(a_i - b_{i-1})$) [104, 103].

The main advantage of the pointwise approach is the simplicity. We can directly reuse existing regression or classification methods. However, it may be difficult to learn good classification (or regression) model for ranking due to the extreme minority of relevant instances [101].

### 2.3.2 Pairwise approach

The pairwise approach is the most popular learning to rank approach. In the pairwise approach, given a query, the input training examples are multiple pairs of documents with a binary preference judgement. The pairwise learning algorithms try to learn a preference function $f$ on the training set by minimizing the pairwise classification loss (i.e., a relevant document is ranked lower than an irrelevant document). In the testing phase, given a query $q$ and two of the testing documents $d_1$ and $d_2$, the preference output $f(d_1, d_2; q) = 1$ means that the document $d_1$ is more relevant than the document $d_2$; $-1$ means less relevant.

Table 2.3: Pairwise approach.

| Method | Input | Output | Loss Function |
|---|---|---|---|
| Pairwise | A pair of documents | Preferences | Pairwise classification loss |

Many work can be categorized into the pairwise approaches, such as RankNet [17], RankBoost [44] and RankSVM [62]. Here, we primarily review the RankSVM algorithm, which is currently used in SEEU.

Similar to [28, 104, 103], RankSVM also represents the learning to rank problem as multiple ordinal regression. Unlike the traditional regression learning, which tries to minimize the mean squared error, RankSVM learns the ranking model by minimizing the pairwise loss in a large-margin optimization framework [53, 62].

$$\begin{array}{c} \min\limits_{w} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{m} \sum_{u,v,y_{u,v}^{(i)}} \zeta_{u,v}^{(i)} \\ subject\ to,\ w^T(x_u^{(i)} - x_v^{(i)}) \geq 1 - \zeta_{u,v}^{(i)}\ ,\ \text{if}\ y_{u,v}^{(i)} = 1 \\ \zeta_{u,v}^{(i)} \geq 0, i = 1, \ldots, m \end{array} \tag{2.15}$$

where $w$ is the model parameter; $C$ is the tradeoff between model complexity and classification error; $\zeta_{u,v}^{(i)}$ is the classification error on the example pair of $x_u$ and $x_v$ in the $i$th training example; $y_{u,v}^{(i)}$ denotes the preference of example $x_u$ over $x_v$ in the $i$th training example.

We can see that the formation of RankSVM is very similar to the training of SVM (see the review in Section 2.1.3). In fact, we can simply treat $x_u - x_v$ as the training examples, and use the classic SVM optimization algorithm to perform binary classification on these examples to learn the model parameter $w$.

It has been shown that minimizing pairwise classification error is equivalent to maximize the lower bound of ranking loss [62]. To make RankSVM scale to very large datasets, recently Joachims [63] proposes the $SVM^{rank}$ algorithm for training RankSVM that proved to have linear time complexity.

### 2.3.3 Listwise approach

The listwise approach directly optimizes the ranking over all the documents associated with a query. Unlike the pointwise and the pairwise approaches, the input of the listwise approach is usually a query and a *list* of documents. Based on the difference of optimization methods, the listwise learning methods can be categorized into two groups, as tabulated in Table 2.4. The first group tries to minimize the loss function defined on the *permutation* of a list of documents, such as ListNet [19]. The second group tries to optimize the surrogate loss of the IR evaluation measure, such as AdaRank [117] and LambdaMART [18]. The surrogate loss is a simplified ranking loss which is mathematically easier to optimize than the complex IR measure, such as NDCG [59].

Table 2.4: Classification of listwise approach.

| Method | Input | Output | Loss Function |
|---|---|---|---|
| Minimization of listwise loss | A list of documents | Permutation | Listwise loss |
| Optimization of IR measure | A list of documents | Ordinal categories | Surrogate loss |

The listwise approach is reported to outperform the pointwise approach and the pairwise approach in the literature [19, 117, 18]. However, the ranking model generated by the listwise approach is usually much more complex than the other approaches. For example, in the recent Yahoo! learning to rank challenge [24], all of the LambdaMART based ranking methods consist of a large number of regression trees. The high model complexity makes it difficult for human interpretation and thus may be impractical in real world application.s

To summarize, in this section, we have briefly reviewed three types of learning to rank approaches, namely the pointwise approach, the pairwise approach and the listwise approach. Due to the simplicity and the efficiency of the pairwise approach, we use the pairwise approach, more specifically, the $SVM^{rank}$ algorithm, to rank documents in each category of the topic hierarchy in SEEU (see Chapter 4).

## 2.4 Topic Modeling

A research area closely related to the work of mining academic topics in universities in SEEU is topic modeling in natural language processing and information retrieval literature. In general, a topic model is a type of mathematical model for discovering abstract (or latent) topics from a collection of documents. The intuitive idea behind topic modeling is that given documents within the same topics, some particular words may appear more frequently than other words. Many topic modeling methods have been proposed recently. The background mathematics of these approaches mainly originates from linear algebra and probabilistic modeling.

In this section, we briefly review three most popular topic modeling algorithms, namely LSA, PLSA and LDA. We will present our topic mining methods in Chapter 6 and compare the results with state-of-the-art topic modeling methods.

## 2.4.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) [34] is a dimensionality reduction technique that projects documents to a lower-dimensional, latent semantic (topic) space where documents with similar topics may be close to each other. By doing so, different terms having similar meaning (synonyms) can be roughly mapped to the same group in the latent space. Thus, it is possible to measure the similarity between pairs of documents even if they do not share any terms. This is very common in text classification and clustering where the high dimensionality of text features usually results in high sparsity of datasets. That is, few terms are shared between most pairs of documents.

LSA models the text as a term-document matrix $X$ where each element $x_{ij}$ represent the occurrence of the term $t_i$ in the document $d_j$. To discover the latent topics behind the corpus, LSA uses Singular Value Decomposition (SVD) to decompose the matrix $X$ into three small matrices as shown in Figure 2.6. The values $\sigma_1, \sigma_2, \ldots, \sigma_l$, are called the singular values, and

$$\begin{array}{ccccccc} X & & U & & \Sigma & & V^T \\ \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} & = & \begin{bmatrix} \begin{bmatrix} \\ u_1 \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ u_l \\ \\ \end{bmatrix} \end{bmatrix} & \bullet & \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{bmatrix} & \bullet & \begin{bmatrix} [ & v_1 & ] \\ & \vdots & \\ [ & v_l & ] \end{bmatrix} \end{array}$$

Figure 2.6: The SVD decomposition of term-document matrix $X$.

$u_1, u_2, \ldots, u_l$ and $v_1, v_2, \ldots, v_l$ are the left and right singular vectors. It turns out that when we select the $k$ largest singular values (i.e., the first k entries along the diagonal of $\sum$) and the corresponding singular vectors $u_1, u_2, \ldots, u_k$ from $U$ and $v_1, v_2, \ldots, v_k$ from $V$, the term-document matrix $X$ can be approximated by $X_k = U_k \sum_k V_k^T$ with minor error. In other words, we actually strip away most trivial dimensions but only keep $k$ important *abstract* dimensions which capture the most variation in $X$. The $k$ remaining vectors in $U_k$ and $V_k^T$ correspond to $k$ hidden topics where terms and documents participate. Based on this, we can project the documents into the semantic space as $D' = \sum_k V_k^T$ and apply any standard clustering algorithm to group documents into topics. In addition, we can also project terms into another latent space as $T' = U_k \sum_k$.

Although LSA can detect synonyms, it may not work well to handle polysemy (i.e., a word may have multiple senses and multiple types of usage in different contexts). The reason is that in SVD decomposition, after we compute the matrix product $U_k \sum_k$, each term $t_i$ is mapped exactly as a single point $t'_i$ in the latent space $T'$, which means that LSA treats the occurrence of each term as the same meaning no matter what document context it appears. This may be unrealistic in a real-world application, such as information retrieval where polysemy terms are very common (e.g., "active learning" in *Computer Science* and *Education*). Probabilistic Latent Semantic Analysis [54], a statistical technique, is proposed to tackle the deficits of LSA.

## 2.4.2 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis (PLSA) [54] is a statistical technique for the analysis of co-occurrence data, such as term-document data. Unlike the Latent Semantic Analysis (LSA) which stems from linear algebra and decomposes the data by Singular Value Decomposition (SVD), PLSA is based on a statistical model, called an aspect model [55] which models the co-occurrence data by associating a latent variable $z$ to each observation (e.g., a term $w$ in a document $d$).

PLSA assumes that the terms in dataset are generated by three steps:

(1) Select a document $d$ with probability $P(d)$.

(2) Pick a latent class $z$ with probability $P(z|d)$.

(3) Generate a word $w$ with probability $P(w|z)$.

This process can be expressed as a joint probability model as

$$P(w,d) = P(d)\sum_{z\in Z} P(w|z)P(z|d) \tag{2.16}$$

where $w$ and $d$ are conditional independent given the latent topic $z$.

Following the maximum likelihood principle, the parameters $P(d)$, $P(w|z)$ and $P(z|d)$ can be learned by maximizing the log-likelihood function

$$\mathcal{L} = \sum_{d\in D}\sum_{w\in W} n(d,w) logP(d,w) \tag{2.17}$$

where $n(d,w)$ is the term frequency of $w$ in document $d$. A standard procedure to learn the parameters in an aspect model is via Expectation Maximization [35].

We can find at least two advantages of PLSA over LSA. Firstly, considering the model interpretability, PLSA has a clear advantage over LSA because PLSA provides a more intuitive definition of the hidden topics. Specifically, in PLSA, for each latent variable (topic) $z$, the top ranked terms by conditional probability $P(w|z)$ give a natural interpretation of the topic meaning for $z$. By contrast, in LSA, the values in the term-topic matrix are not normalized and may even contain negative values which may be difficult to interpret. Secondly, both LSA and PLSA associate terms into latent topics and thus can handle synonymy. However, LSA may not work well to handle polysemy, as terms in LSA are mapped into a single point in the semantic space. For PLSA, given a term $w$, the different values of conditional probabilities $P(w|z)$ for different topics $z$ give a natural explanation of polysemy.

Although PLSA has some advantages over LSA, it still has several weaknesses. A major difficulty of PLSA is to be prone to overfitting because the number of parameters of PLSA grows linearly with the number of training documents [11]. Specifically, given a text dataset with $V$ terms and $M$ documents, training PLSA with $k$ latent variables (topics) requires to learn $kV + kM$ parameters, which are linearly growing with $M$, which in turn is usually very large. The large number of parameters is due to the $k$ latent variables explicitly linked to the training documents ($P(z|d)$ in Equation 2.16). To avoid the overfitting problem, Latent Dirichlet Allocation (LDA) [11] proposes a different generative process that reduces the parameters to $kV + k$ which does not grow with the number of documents.

### 2.4.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [11] is a statistical learning algorithm for automatically detecting topics in documents. Similar to PLSA, LDA also views each document as a mixture of various topics. The difference is that LDA does not model the topic mixture in documents explicitly. LDA treats the topic mixture as a k-parameter hidden random variable with a Dirichlet prior (i.e., a distribution on multinomials). Specifically, for each document in the collection, LDA assumes the following generative process:

(1) Randomly choose a per-document topic distribution based on the Dirichlet prior.

(2) Randomly choose a per-topic word distribution for each topic based on another Dirichlet prior.

(3) For each word in the document

(a) Randomly choose a topic from the distribution over topics.

(b) Randomly choose a word from the distribution over the vocabulary given the sampled topic.

Mathematically, the LDA model can be defined with the following notations. We denote the Dirichlet priors on the per-document topic distribution and per-topic word distribution as $\alpha$ and $\beta$. The topic distribution for document $i$ is $\theta_i$. The word distribution for topic $k$ over vocabulary is denoted as $\phi_k$. $z_{j,t}$ is the topic for the $t$th word in document $j$ and $w_{j,t}$ is the $t$th word in document $j$. Based on the generation process, we can write the joint probability of all known and hidden variables as

$$P(w, z, \theta, \phi; \alpha, \beta) = \prod_{i=1}^{K} P(\phi_i; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N} P(z_{j,t}|\theta_j) P(w_{j,t}|\phi_{Z_{j,t}}) \tag{2.18}$$

Estimating parameters of LDA by exactly maximizing the likelihood of the whole data collection is intractable. A popular solution to this problem is to use approximation methods, such as variational algorithms [11] and Gibbs sampling [46].

As a statistical graphical model, LDA shares similar advantages of PLSA over traditional LSA method. Moreover, LDA is reported to be less likely overfitting than PLSA in empirical study. The Dirichlet priors $\alpha$ and $\beta$ are assumed to be given. The parameters that LDA needs to learn are $P(z|\theta)$ and $P(w|\phi)$. Thus, there are totally $k + kV$ parameters which are much smaller than PLSA.

We have reviewed three most popular topic modeling approaches in the literature. These approaches have shown promising results in discovering hidden topics from text dataset. However, we argue that these approaches may not be suitable for SEEU. There are three main reasons.

1. **Unpredictable topics**. An important feature of topic modeling is to discover hidden topics from text dataset. However, all of these approaches are unsupervised learning

methods. Without supervision information, the topics learned by these approaches may be unpredictable and even unmeaningful and thus confuse users. In addition, although some hierarchical extensions, such as hierarchical PLSA [116] and hierarchical LDA [10] have been proposed, the hierarchical topic structures generated by these methods are still unpredictable.

2. **Unknown topic names**. All of the three approaches represent topics in a mathematical form, such as numerical vector in LSA or word distribution in PLSA and LDA. A serious problem of such methods is the unknown topic names. Simply grouping the results by topics without labeling the name of the topics could cause a bad user experience. Moreover, according to [21], assigning meaningful labels to the topics (clusters) is a very difficult task.

3. **Computational inefficiency** The three approaches stem from solid mathematical background, such as SVD decomposition in LSA and graphical model in statistics. However, when we apply these approaches to large-scale webpage dataset, these methods may not be efficient. For example, given a new webpage, PLSA needs to rerun the EM algorithm on the entire text dataset to infer its topics. This could be too slow in real-world search engines.

Due to the weakness we discussed above, we prefer to use supervised text classification methods for mining topics in universities. We will compare the topic mining performance between supervised text classification methods and topic modeling methods in Chapter 6.

## 2.5 Summary

In this chapter, we have reviewed three different machine learning methods for classifying and searching documents. They are the supervised text classification methods, the learning to rank methods and the unsupervised topic modeling methods. For classifying documents in SEEU, due to the drawbacks of the unsupervised topic modeling methods, we prefer to use the supervised text classification methods, more specifically, the hierarchical text classification methods for hierarchical topic classification. For ranking documents in SEEU, we will use the pairwise learning to rank approach to rank documents in each category of the topic hierarchy.

# Chapter 3

# Effective Hierarchical Webpage Classification

In this chapter, we study the hierarchical webpage classification problem. A major challenge in SEEU is to automatically classify a massive number of webpages into a topic hierarchy. An effective hierarchial classification system is very important for SEEU. To deal with the challenges of learning large-scale webpage datasets, we firstly propose an efficient webpage feature extraction tool based on MapReduce. Secondly, we develop a parallel hierarchical SVM classifier for effective webpage classification. With extensive experiments on the well-known ODP (Open Directory Project) dataset, we empirically demonstrate that our hierarchical classification system is very effective and it outperforms the traditional flat classification approach significantly.

The rest of this chapter is organized as follows. In Section 3.1, we describe the webpage feature extraction tool for hierarchical webpage classification. In Section 3.2, we discuss the algorithm to learn hierarchical classifiers. Section 3.3 reports the experimental results on the ODP (Open Directory Project) dataset. The last section contains a summary of this chapter.

The implementation of the hierarchical classification system in Section 3.2 was in collaboration with Da Kuang and Dr. Charles Ling. We jointly published this work in the *Proceeding of the 22nd International Joint Conference on Artificial Intelligence* (IJCAI 2011) [67].

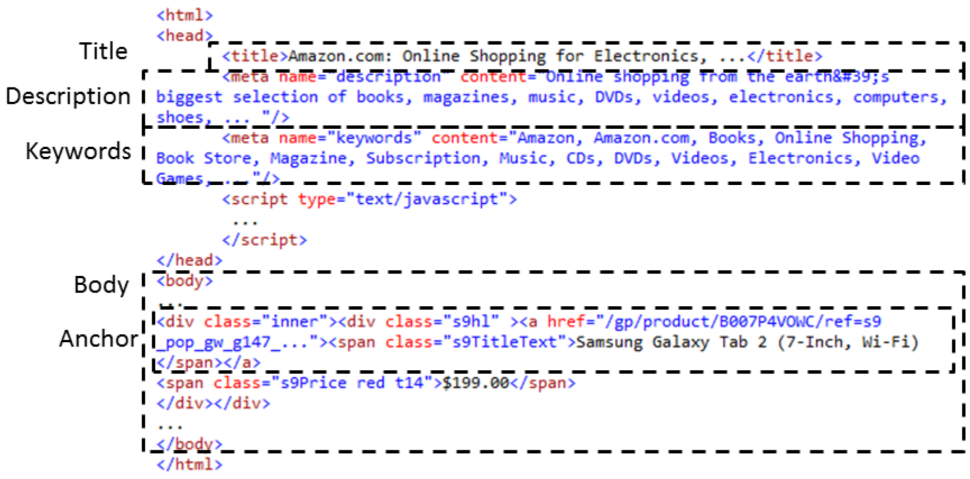
## 3.1 Webpage Feature Extraction

When we apply text classification algorithms on real-world webpage datasets, the first thing we need to do is to extract good text features from webpages. In this section, we describe the webpage features for learning hierarchical classification models. To deal with the challenges of extracting text features from a large-scale webpage dataset (e.g., one million webpages in the ODP dataset), we develop a distributed feature extraction tool based on the popular Hadoop MapReduce platform[1].

[1]The website of Hadoop project is `http://hadoop.apache.org/`.

### 3.1.1 Webpage Features

To extract text features from webpages, it is important to consider the tag structure of the HTML source code. We use a webpage (see Figure 3.1) collected from Amazon to describe features extracted from HTML source. As we can see, HTML source code usually consists of a head part and a body part. Firstly, inside the head part, we extract the title as well as two meta texts, i.e., description and keywords. The three text features are very important as they describe the theme (such as shopping in Amazon) and the content (e.g., Books, Music and Games) of the webpage. Secondly, for the body part, we simply treat it like plain text by removing all the HTML tags. In this thesis, we do not consider the primary HTML tags such as head (<h>), paragraph (<p>) and section (<div>), because in most websites, these tags are oriented toward visualization rather than semantics [91]. However, for anchor tags (<a>), we cannot simply discard them because the text inside anchor tags is usually very relevant to the *pointed* webpage [14]. Therefore, we also use the anchor text feature for the *pointed* webpage. In addition, we extract text from the webpage URL (not the links inside the anchor tags) because the URL of a webpage also contains useful information.[2] Thus in total, we use six text features for webpage classification. They are tabulated in Table 3.1.



```
<html>
<head>
      <title>Amazon.com: Online Shopping for Electronics, ...</title>
      <meta name="description" content="Online shopping from the earth&#39;s
biggest selection of books, magazines, music, DVDs, videos, electronics, computers,
shoes, ... "/>
      <meta name="keywords" content="Amazon, Amazon.com, Books, Online Shopping,
Book Store, Magazine, Subscription, Music, CDs, DVDs, Videos, Electronics, Video
Games, ... "/>
      <script type="text/javascript">
       ...
      </script>
</head>
<body>
...
<div class="inner"><div class="s9hl" ><a href="/gp/product/B007P4VOWC/ref=s9
_pop_gw_g147_..."><span class="s9TitleText">Samsung Galaxy Tab 2 (7-Inch, Wi-Fi)
</span></a>
<span class="s9Price red t14">$199.00</span>
</div></div>
...
</body>
</html>
```

Figure 3.1: A simplified HTML source code from Amazon home page.

Table 3.1: The six text features of a webpage. They are URL, title, description, keywords, body and the anchor text.

| ID | Description |
|---|---|
| 1 | URL |
| 2 | title |
| 3 | description in meta tag |
| 4 | keywords in meta tag |
| 5 | body |
| 6 | anchor text from inbound hyperlinks |

[2]For example, the link `http://www.amazon.com/books-used-books-textbooks/...` points to the book department of Amazon. It contains useful words including *books*, *used books* and *textbooks*.

### 3.1.2 MapReduce based Feature Extraction

A webpage dataset is usually very large. An efficient feature extraction implementation is non-trivial. Simple sequential scanning over the entire webgraph could be too slow for a large-scale dataset. In this thesis, we build the feature extraction tool based on the popular Hadoop MapReduce platform.

We briefly describe the MapReduce programming model. MapReduce is a parallel programming model for processing large-scale datasets. In a MapReduce development, the complex low-level system programming, such as data communication, load balancing and fault tolerance are taken over by the MapReduce platform. Developers only need to focus on the implementation of high-level algorithms by using the simple *map* and *reduce* functions [32]. In a typical MapReduce program, the main computation procedure can be implemented as a series of data manipulation on key-value pairs. Specifically, the main program splits the problem into many small subproblems. For each subproblem, the MapReduce platform launches a *map* function that processes the subproblem and outputs intermediate results as a list of key-value pairs. When all the *map* functions are finished, the MapReduce platform reorganizes all the intermediate key-value pairs into many value lists of identical keys. For each value list, a *reduce* function will be launched to process it and output a single key-value pair as the final results.

We implement the feature extraction tool based on the MapReduce programming model. Figure 3.2 shows an example of MapReduce pseudo-code for extracting anchor text. It contains a *map* function and a *reduce* function. The entire webpage dataset is split into many webpages by URLs. For each webpage, the *map* function extracts and emits pairs of ("hyperlink", "anchor text") from HTML source code. After all the *map* functions are finished, the *reduce* function combines the list of anchor text for a URL to form the final anchor feature for a webpage as ("hyperlink", "merged anchor text"). The code to extract the other text features is simpler than extracting anchor text. They only have a *map* function which just extracts the in-page text.

```
map(String url, String html):
    //url: the URL of a webpage
    //html: the HTML source
    for each hyperlink in html:
        EmitIntermediate(hyperlink, anchorText);

reduce(String hyperlink, Iterator anchorTextList):
    //hyperlink: the URL of a webpage
    //anchorTextList: a list of anchorText
    String anchorFeature = "";
    for each anchorText in anchorTextList:
        anchorFeature += " " + anchorText;
    Emit(hyperlink, anchorFeature);
```

Figure 3.2: The MapReduce pseudo-code for extracting anchor text feature.

## 3.2 Hierarchical Classification

A major challenge in building search engines with hierarchies is to automatically classify a massive number of webpages into various hierarchies. In this section, we use the top-down hierarchical classification approach (see review in Chapter 2.2) for training hierarchical classification system.

### Top-down Hierarchical Classification Algorithm

Given a predefined topic hierarchy $\mathcal{H}$, we build one binary classifier on each category of $\mathcal{H}$. The positive and negative examples for each category are selected *locally* [106]. Specifically, for each category $c \in \mathcal{H}$, let $\uparrow(c)$ denote the direct parent category of $c$ and $E(c)$ denote the examples belonging to $c$. Then the positive training examples of $c$ can be defined as $Tr_+(c) = E(c)$[3] and the negative training examples of $c$ can be defined as $Tr_-(c) = Tr_+(\uparrow(c)) - Tr_+(c)$. As the negative training examples are only selected from the positive examples of the parent categories (locally), the classifier would not suffer from the serious imbalance problem [58].

In prediction phase, the top-down approach classifies testing examples in a top-down manner. Specifically, given a testing example, the classifiers at the top level firstly output its prediction probabilities. After that, at each category, if the probability of the example is larger than a threshold (e.g., 0.5), the example will be *recursively* pushed down to its lower-level child categories for further prediction until reaching the leaf categories.

Tuning prediction thresholds in the top-down approach is very important. Too small prediction thresholds will cause examples wrongly predicted into deeper categories while too high prediction thresholds will cause examples to be blocked at top level. To tune the optimal prediction thresholds, we use the popular SCut [121, 122] algorithm. Its basic idea is to tune the threshold of each category on a validation set (e.g., 10% training examples) until optimal performance of the classifier is obtained for that category (on the validation set). In our classification system, we tune the prediction threshold of each category in the set of $\{0.3, 0.4, 0.5, 0.6, 0.7\}$.

We use linear Support Vector Machine (SVM) as the base classifier, because the high dimensionality of text data usually results in the dataset being linearly separable [112]. The popular LIBLINEAR [41] package is used in our implementation. To obtain prediction probabilities from SVM, we implement the famous Platt's calibration [88] to output calibrated probabilities by sigmoid function,

$$P(y = 1|x) = \frac{1}{1 + \exp(A \cdot f(x) + B)} \tag{3.1}$$

where $f(x)$ is the SVM output of example $x$; $A$ and $B$ are the parameters learned by Platt's calibration.

### Feature Preprocessing and Selection

We use the bag-of-words model to represent the text data. Each document is treated as a vector of word features with TF·IDF weighting. Before the construction of the TF·IDF word vector

[3]For the root category, all the training examples are positive.

of each document, we need to remove stop words (e.g., *the*, *a*, *and*) and rare words (e.g., words occurring in less than three documents) as they are not useful for learning a classification model. In addition, as we are mainly learning the English documents, words with the same meaning can appear in different forms, such as *learning*, *learn* and *learns*. We use the Porter Stemming [89] algorithm to normalize words to their stem (e.g., *learning* → *learn*).

The feature space in the text classification dataset is usually very large. Training SVM classifiers on hundreds of thousands of documents in a high dimensional feature space requires a great number of computational resources. To reduce computational cost, we use the DF (Document Frequency) [42] feature selection algorithm to select a small portion of relevant features. Specifically, the DF algorithm assigns a score to each word in the training data. The DF score is calculated as the number of documents containing this word. The top ranked features by the DF scores can be used as useful features for classification. Although the DF algorithm is very simple, it has been shown that the the DF algorithm is a reliable measure for selecting informative features [123].

## Time Complexity

We analyze the time complexity of training the hierarchical SVM classifiers. Let us consider a dataset that contains $m$ examples distributed in a hierarchy $\mathcal{H}$ of depth $d$. The feature size of the dataset is $n$.

We use the top-down approach to train a linear SVM classifier at each category of $\mathcal{H}$. It has been proved that the time complexity of training a linear SVM grows linearly with the number of training examples and features [63]. The time complexity of the DF feature selection is also linear as we only need to scan the training set once to compute the DF scores for all words. For the SCut algorithm, estimating the prediction loss for each threshold value requires classifying all the examples in the validation set. As we only compare a few threshold values on a small portion of training examples (e.g., 10% of the training set), the complexity of SCut can be approximately considered as linear. Thus, the total time complexity of hierarchical SVM classifiers is the sum of the complexity of training all the linear SVM classifiers,

$$T_{train} = \sum_{i=1}^{d} \sum_{j=1}^{|\mathcal{H}_i|} O(m_{ij} n) \tag{3.2}$$

where $|\mathcal{H}_i|$ is the number of categories at the depth $i$; $m_{ij}$ is the number of training examples in the jth category at depth $i$.

To simplify the derivation of time complexity in a hierarchy, we make two assumptions here.

- Assumption 1. We assume that each intermediate category has a unified branching factor $b$ (i.e., a fixed number of child categories).

- Assumption 2. We assume that we are facing the mandatory-leaf node classification [9, 106] where each example in the training set is categorized on a single path from the root to a leaf category.

**Theorem 3.2.1** (Complexity of Hierarchical Classification) *Given a dataset containing m examples distributed in a hierarchy of depth d with n features, the time complexity of training hierarchical SVM classifiers is*

$$T_{train} = \sum_{i=1}^{d} \sum_{j=1}^{|\mathcal{H}_i|} (m_{ij} n) = dbO(mn)$$

**Proof** We use mathematical induction to prove this. Given a hierarchy of depth $d$, we want to prove $T_{train}(d) = dbO(mn)$.

For a hierarchy of depth 1, we train $b$ SVM classifiers on the entire training set. Thus, we can derive $T_{train}(1) = bO(mn)$. We suppose that $T_{train}(d)$ is true for a hierarchy of depth $d$. We need to prove that $T_{train}(d+1)$ is also true for a hierarchy of depth $d+1$.

Given a hierarchy of depth $d+1$, we know that the complexity of training hierarchical SVM classifiers on the top $d$ levels is $dbO(mn)$. Consider a category $j$ at depth $d$, we use $E_{dj}$ to denote the positive examples belonging to it. According to the top-down approach, for each of its subcategories at depth $d+1$, the number of training examples is exactly $E_{dj}$ (i.e., positive examples from the parent category). Based to Assumption 1, we can derive that the total time complexity for its child categories is $bO(E_{dj}n)$. Therefore, the total time complexity at depth $d+1$ is $b\sum_{j=1}^{|\mathcal{H}_d|} O(E_{dj}n)$. According to Assumption 2, each training example must pass only once through a category at depth $d$. Thus, the total examples passing through depth $d$ is $\sum_{j=1}^{|\mathcal{H}_d|} E_{dj} = m$. Therefore, $b\sum_{j=1}^{|\mathcal{H}_d|} O(E_{dj}n) = bO(mn)$. Eventually, we prove that the time complexity of training hierarchical SVM classifiers for a hierarchy of depth $d+1$ is $T_{train}(d+1) = dbO(mn) + bO(mn) = (d+1)bO(mn)$. ∎

Next, we derive the time complexity of the flat approach for comparison. In the flat approach, we ignore the hierarchy structure and train SVM classifiers at all the leaf categories. For each leaf category, a SVM classifier is trained to distinguish that category from all the other leaf categories. Specifically, the positive training examples of each leaf category $c$ can be defined as $Tr_+(c) = E(c)$ and the negative training examples of $c$ can be defined as $Tr_-(c) = D - Tr_+(c)$ where $D$ is the entire training set. Thus, the time complexity of the flat approach can be simply derived as $b^d O(mn)$ ($b^d$ is the total number of leaf categories in $\mathcal{H}$). We can see that the hierarchical SVM classifiers are exponentially faster than the flat approach. For example, consider a hierarchy of four levels with branching factor of five. The flat approach has to train 625 ($5^4$ leaf categories at the 4th level) full SVM classifiers (on the entire dataset) while the time complexity of the hierarchical SVM classifiers is just equivalent to train 20 ($= 4 \times 5$) full SVM classifiers.

We study the time complexity of predicting a testing example. Given a testing example predicted from the root to a leaf category, the hierarchical SVM classifiers only use the SVM classifiers attached to the categories of the prediction path. In other words, $b$ classifiers are used at each level of the prediction path. Thus, its time complexity is

$$T_{test} = dbO(n) \tag{3.3}$$

For the flat approach, as each classifier at a leaf category must be used, its time complexity is $b^d O(n)$. We can see that the hierarchical SVM classifiers approach is again far more efficient than the flat approach.

**Parallel Speed Up**

We have proved that the hierarchical SVM classification algorithm is much faster than the flat approach. However, in a real world application with millions of training examples in large hierarchies, it could be still slow to sequentiality train many SVM classifiers.

Although we can also use MapReduce for parallelization, we prefer to use OpenMPI[4], a popular High Performance Computing library, because training many SVMs is a computationally intensive task rather than data intensive task where MapReduce is especially suited for [78]. Thus, similar to [81], we also develop a parallel machine learning system on a small cluster of PCs.[5] The MPI pseudo-code of training hierarchical SVM classifiers is presented in Figure 3.3. It is based on the classic Master/slave parallel computing model. Simply speaking, each worker requests a job (category) from the master, and trains the SVM classifier for that category. This process is repeated until all the jobs are consumed. In an ideal case, given a cluster of $p$ processors, the time complexity of training hierarchical SVM classifiers can be further reduced to

$$T_{train} = p^{-1} dbO(mn) \tag{3.4}$$

```
if rank != 0:
    /*Worker module*/
    while true:
        //request a category from the master
        MPI_Send(req, 0);
        MPI_Recv(category, 0, stat);
        //quit if the category is empty
        if category=Null:
            return;
        //Top-down building training set
        TrSet = Localize(Data, category);
        //Do feature selection on the training set
        TrFSSet = FeatureSelection(TrSet);
        //Train a SVM model
        SVM = TrainSVM(TrFSSet);
        //Use SCut to tune the optimal threshold
        Threshold = SCut(TrFSSet);
else:
    /*Master module (rank=0)*/
    //schedule categories to workers
    for category in Hierarchy:
        MPI_Recv(req, MPI_ANY_SOURCE, stat);
        MPI_Send(category, stat.source);
    //All jobs are finished, tell each worker to stop
    for worker in pool:
        MPI_Recv(req, MPI_ANY_SOURCE, stat);
        MPI_Send(Null, stat.source);
```

Figure 3.3: The MPI pseudo-code for training hierarchical SVM classifiers.

[4] `http://www.open-mpi.org`

[5] It consists of four quad-core PCs each with four 2.4 GHz CPU cores and 3GB memory.

## 3.3 Hierarchical Classification Experiments

In this section, we describe the experimental dataset and conduct an empirical study on the classification performance of hierarchical classification methods.

### 3.3.1 Experimental Dataset

We use the well-known ODP (Open Directory Project) hierarchy as our experimental data. ODP is a human-edited web directory where human experts organize millions of webpages into a complex topic hierarchy with a maximal depth of 14. We extract a meaningful *4-level* topic hierarchy from the original hierarchy of ODP, since it is difficult to train highly accurate classifiers at the deep levels of the hierarchy [81]. Besides, a very deep hierarchy is not convenient for users to browse. To have a broad coverage of common topics, we select 11 out of the total 16 top categories in the ODP hierarchy. They are *Arts*, *Business*, *Computers*, *Health*, *Home*, *Recreation*, *Reference*, *Science*, *Shopping*, *Society* and *Sports*. After the data collection and cleaning, we obtain 1,047,560 webpages distributed in 662 categories.[6] Here, we report the statistical information of our topic hierarchy at each level in Table 3.2.

Table 3.2: The statistical information of the topic hierarchy at each level. $c$ is the total number of categories. $\overline{r}$ is the average positive class ratio over all documents. $\overline{n}$ is the average number of examples. $\overline{f}$ is the average number of distinct words.

| Level | $c$ | $\overline{r}$ | $\overline{n}$ | $\overline{f}$ |
|---|---|---|---|---|
| 1 | 11 | 0.094 | 1,047,560 | 755,718 |
| 2 | 99 | 0.091 | 104,039 | 254,606 |
| 3 | 499 | 0.103 | 11,026 | 66,565 |
| 4 | 53 | 0.106 | 7,715 | 68,654 |

### 3.3.2 Evaluation

To evaluate our classification system, we conduct five-fold cross validation experiments on the ODP dataset. Specifically, we randomly split the dataset into five subsets (folds) with equal size. At each round of cross validation, four folds are used for training (with 838,048 examples) and the rest is retained for testing (with 209,512 examples). When training SVM classifiers, we use the default parameters (i.e. $C = 0$) of LIBLINEAR. In the testing phase, we report the average F1-score [121] at each level of the hierarchy.

#### Comparing the Top-down Approach and the Flat Approach

We firstly compare the performance of the hierarchical SVMs (the top-down approach) with the traditional flat SVMs (the flat approach). For both approaches, we set the feature number to 50,000, and set the prediction thresholds for all categories as 0.5. The SCut algorithm is not used for the top-down approach.

[6] Although we only choose four-level categories, the webpages in deep levels of the ODP hierarchy are popped up and collected in the categories of the fourth level.

Figure 3.4 presents the results. We can see that the top-down approach has a clear advantage over the flat approach. Generally speaking, the deeper the level is, the more significant improvement the top-down approach is. The largest improvement can be observed at the 3rd and the 4th levels. We find that for the flat approach at these two levels, the average positive class ratios are less than 0.001, much more skewed than the top-down approach (i.e., about 0.1). With serious imbalanced class distribution, it is very difficult to learn a good classification model for the flat approach. In addition, we notice that the flat approach is far slower than the top-down approach. On average, it takes about ten hours to finish training while the top-down approach uses less than one hour. For the large number (575) of leaf categories in the ODP hierarchy, the flat approach trains SVM classifiers on the *entire* training set while the top-down approach trains SVM classifiers only on *exponentially smaller* portion of examples.

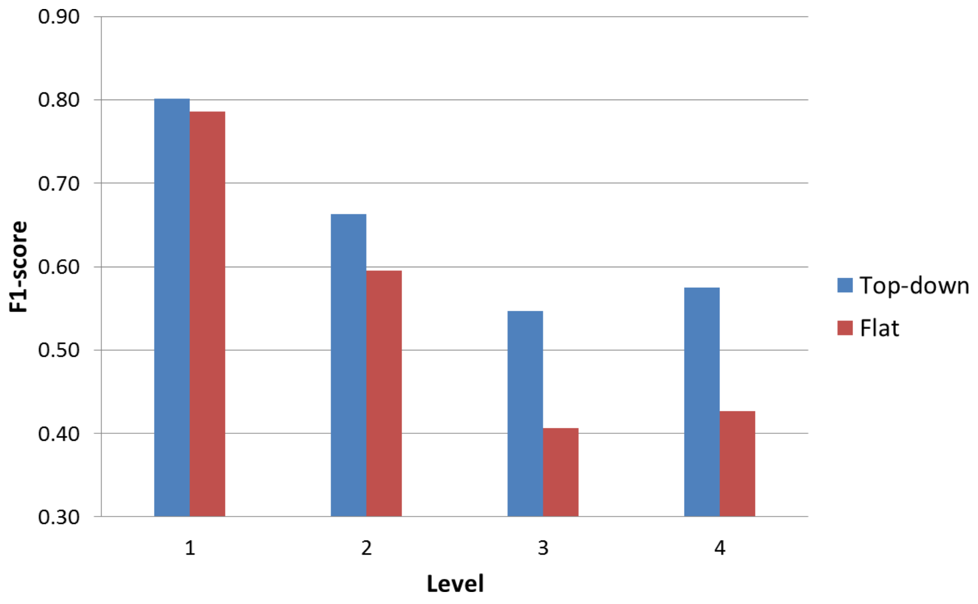


Figure 3.4: The average F1-score at different levels by the top-down approach and the flat approach.

### Optimal Thresholds in Hierarchical SVM classifiers

Secondly, we study the thresholds in our hierarchical SVM classifiers. In this experiment, we compare the SCut method with the fixed threshold value 0.5, a common prediction threshold in text classification. For the SCut method, we tune the threshold of each category on a validation set (e.g., 10% randomly sampled training examples) until the optimal performance of the classifier is obtained for that category on the validation set.

We repeat the five-fold cross validation experiments for the hierarchical SVM classifiers, and report the results in Figure 3.5. We can see that the SCut method consistently outperforms the fixed threshold method at all levels. It means that in hierarchical classification, it is important to tune the prediction thresholds to achieve high classification performance.

It is also worthy to analyze the distribution of the tuned thresholds by the SCut method. We plot the distribution of optimal thresholds tuned by the SCut method at all levels in Figure

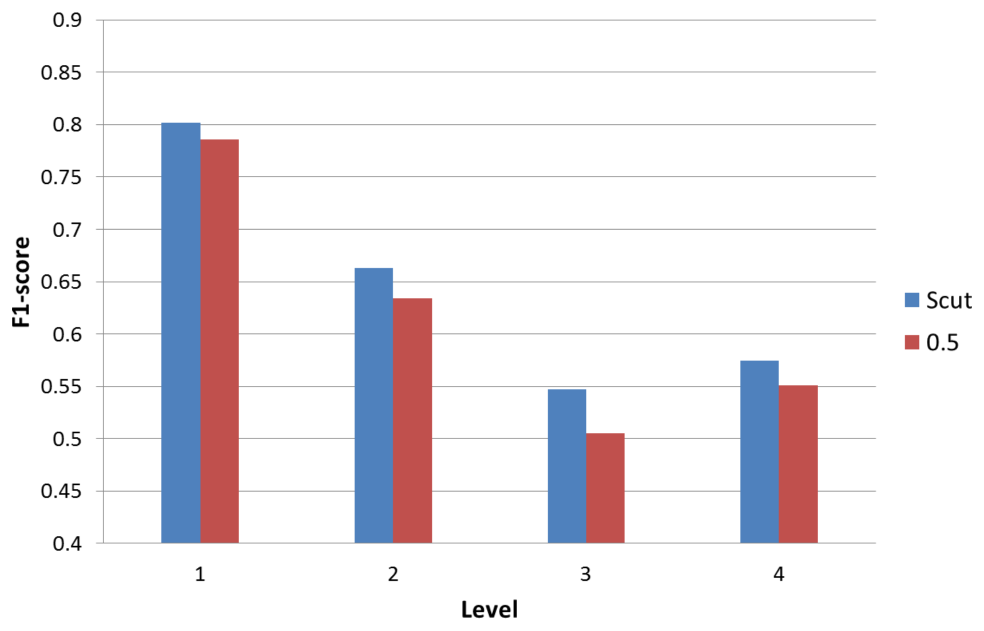


Figure 3.5: The average F1-score at different levels by the SCut thresholds and the fixed threshold 0.5 for the top-down approach.

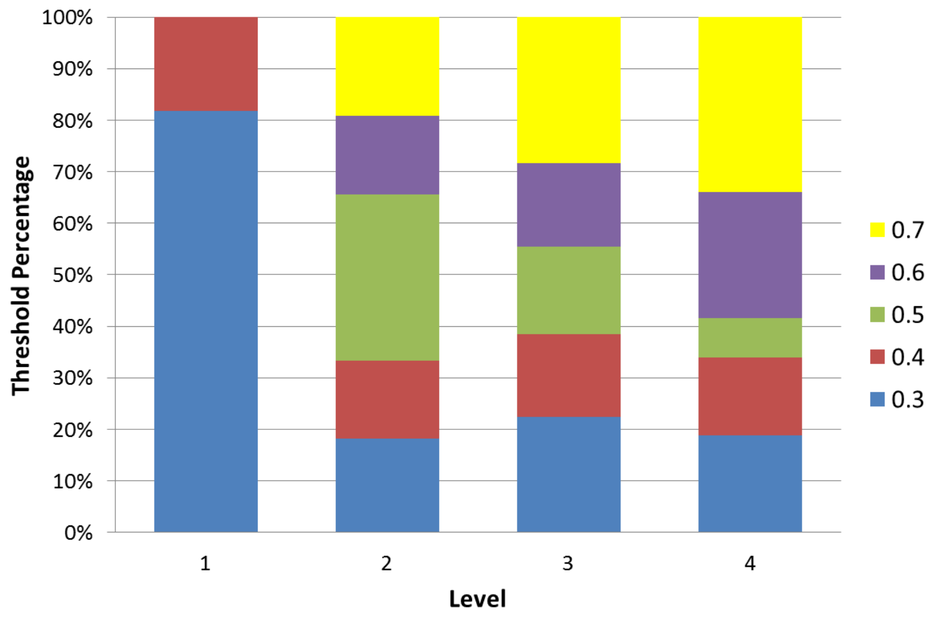


Figure 3.6: The optimal threshold distribution at different levels by the SCut method.

3.6. We can see that at the top two levels, most of the threshold values are less than 0.5 while at the 3rd and the 4th levels, larger thresholds (i.e., 0.6 and 0.7) are more often used. This is reasonable. For the top level, a small threshold value can push more examples down to the lower levels, and reduces the class imbalance problem in deeper levels. Thus, the recall at deeper levels can be improved. In addition, this also helps to resolve the blocking (false negative) problem [109] in hierarchical classification. One may say that this may introduce false positive examples into the deeper levels. To reduce such errors propagated from the top level, the SCut method increases the thresholds for some categories at the deeper levels. This helps to maintain the precision. As the SCut method can improve both precision and recall for the large number of deeper-level categories, the overall F1-score will be eventually improved.

### Effect of Feature Number in the Hierarchical SVM classifiers

Next, we study the effect of different feature numbers on the classification performance of the hierarchical SVM classifiers. The feature numbers used in this experiment are 5,000, 10,000, 50,000 and 100,000. Figure 3.7 plots the F1-score at all levels under different feature numbers. We can see that increasing the number of selected features can improve F1-score consistently at all levels. This is as we expected as SVM learns more accurately with a large number of relevant features [60].

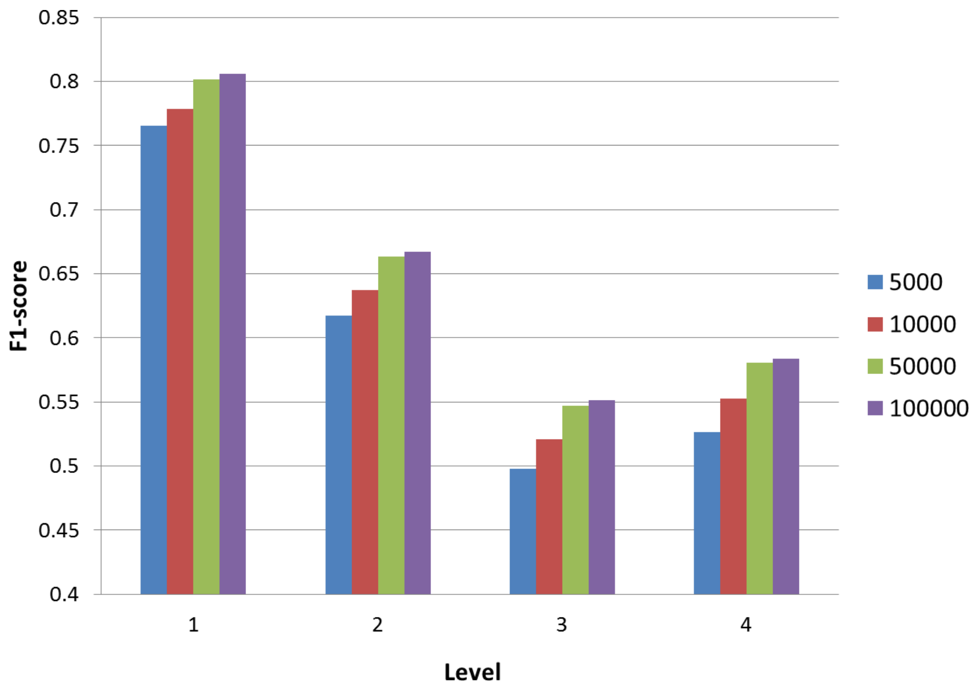


Figure 3.7: The average F1-score at different levels by different feature numbers.

We also analyze the correlation between the running time and different feature numbers for the hierarchical SVM classifiers. Figure 3.8 plots the average training time and testing time under different feature numbers. We can see that the training time steadily increases along with the growth of feature numbers. This is because a larger number of features will make the SVM learning algorithm consume more computational resource and thus become slower to converge. However, for testing time, the correlation between feature number and running time is not clear. The running time is mainly decided by the number of examples due to the sparsity of features in text documents [41]. In fact, although the ODP dataset has more than one million distinct words, we find that the average number of distinct words per document is just 198. From this analysis, we believe that 50,000 features are good enough for the hierarchical SVM classifiers when applied for a hierarchy of medium size (e.g., with several hundred categories).

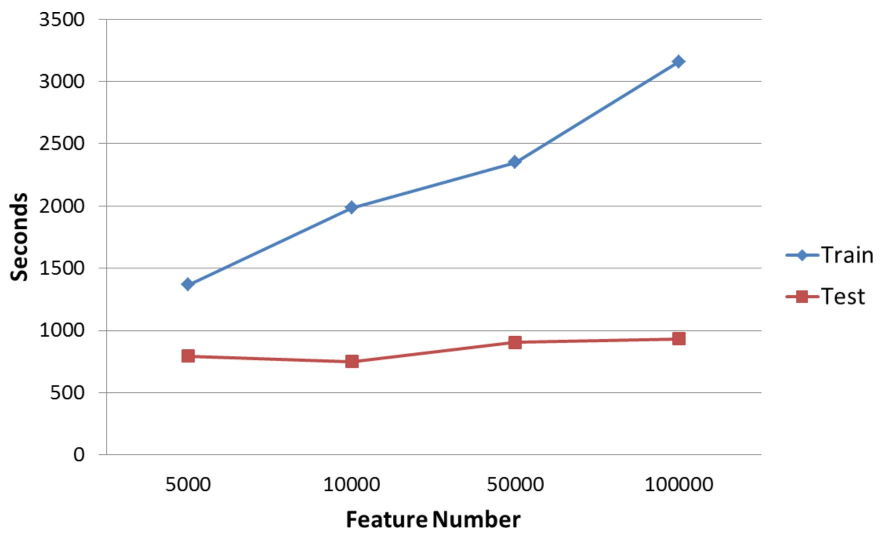


Figure 3.8: The average training and testing time of the hierarchical SVM classifiers under different number of features.

### Throughput of the Hierarchical SVM classifiers

Finally, we analyze the efficiency of the hierarchical SVM classifiers. From Figure 3.8, we can see that the training and testing of the hierarchical SVM classifiers are quite efficient. For 50,000 features, the average training time at one fold of cross validation (with 838,048 examples) is 2,350 seconds. The average testing time on one fold (209,512 examples) is 904 seconds.[7] The throughput (number of documents classified per second) of our classification system is 231 (209512/904), which is quite good considering only a small cluster of PCs is used. Since both the training and testing processes can be performed off-line, we believe that the scalability of the hierarchical classification for a large search engines is feasible.

## 3.4 Summary

In this chapter, we develop an effective hierarchical classification system for large-scale webpage classification in a topic hierarchy. According to our experimental results on the well-known ODP dataset, we empirically demonstrate that our hierarchical classification system is very effective and outperforms the traditional flat classification approaches significantly.

[7]In our previous paper [67], we do not count the time of disk I/O and network I/O. This could be unrealistic for real world systems. In this thesis, we count the total execution time (including disk I/O and network I/O) of running a MPI job on our cluster.

# Chapter 4

# ERIC: Enhanced Ranking by hIerarchical Classification

Traditional keywords-based web search engines rank documents only based on the text matching between a query of keywords and the indexed documents, and page importance metrics. However, in search engines with hierarchies, we also need to consider the topic relevance in hierarchies. In this chapter, we study the problem of integrating hierarchical classification into keywords-based search engines. We propose a novel ranking framework, called ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), that improves the ranking quality of a search engine by hierarchical classification. With extensive experiments on four TREC (Text REtrieval Conference) web search datasets, we empirically demonstrate that our ranking framework with hierarchical classification outperforms the flat keywords-based search methods significantly. To the best of knowledge, this is the first work that evaluates the performance of ranking in hierarchies on the TREC datasets with satisfying results.

The learning to rank experiments in Section 4.3 are done with Dr. Charles Ling and Dr. Huaimin Wang. This work was included in the submission to the *IEEE Transactions on Knowledge and Data Engineering* (IEEE TKDE) [75].

## 4.1 Introduction

In traditional keywords-based web search engines, searching documents is usually considered as a similarity match problem between a query of keywords and the indexed documents, boosted by page importance metrics (such as PageRank [85]). Usually, the matching algorithm is based on the TF·IDF weighting with human crafted parameters, such as the popular BM25 [94] ranking methods adopted by most search platforms.

As user queries are usually very short and ambiguous [105], a simple keywords match may fail to capture the true similarity between queries and the index documents. For example, if users want to find information about "active learning" (a research field in *Computer Science*), they could search keywords "active learning" in a flat search engine, such as Google. Due to the ambiguity of the short phrase "active learning", most of the top ten results are related to *Education* research. If we combine more keywords, such as "Computer Science", webpages about "active learning" without the words "Computer Science" may be filtered out, and thus, users may miss some important webpages. However, in a search engine with hierarchies, such as SEEU, if users select the category “Computer Science” in the topic hierarchy while searching “active learning” as the keywords, the top ten ranked results will be exclusively related to *Computer Science* research and the results without the words “Computer Science” will still remain.

The major challenge of integrating hierarchies into a search engine is to classify and rank a large number of webpages into hierarchies. In previous Chapters, we have studied the problem of large-scale hierarchical webpage classification. In this chapter, we study the problem of using hierarchical classification to improve the ranking performance of search engines. We propose a novel ranking framework, called ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification) that integrates the hierarchical classification probabilities into the ranking system of search engines with hierarchies. In this framework, we develop several ranking features to capture text similarity between queries and documents, as well as topic correlation between user selected categories and the indexed documents, and integrate them into a learning to rank algorithm. With extensive experiments on four TREC (Text REtrieval Conference) web search datasets, we empirically demonstrate that by seamlessly integrating hierarchical text classification and ranking methods, our framework can boost search engine’s ranking quality significantly. To the best of knowledge, this is the first work that evaluates the performance of ranking in hierarchies on the TREC datasets with satisfying results.

The rest of this chapter is organized as follows. In Section 4.2, we discuss the challenges of integrating hierarchical classification into the ranking, and propose a novel framework to tackle the challenges encountered. In Section 4.3, we conduct extensive experiments to evaluate the effectiveness of our approach, compared with the state-of-the-art flat ranking methods. The last section presents the summary.

## 4.2 Integrating Hierarchical Classification into Ranking

When we integrate hierarchical classification into search engines with hierarchies, two challenges need to be solved.

- The first challenge is how to define a commonly accepted topic hierarchy for the application domain and effectively classify the massive number of webpages into the hierarchy.
- The second challenge is how to design effective features for a ranking system. Improper features may even degrade the performance of a ranking system.

In this section, we will propose a novel rank framework to tackle the two challenges.

### 4.2.1 General Framework

We propose a novel ranking framework called ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification) for search engines with hierarchies. Figure 4.1 presents the main idea of our framework. Simply speaking, our framework can be considered as a feature processing network which transforms each query-document pair into three different types of features for a ranking system. The key and novel idea of our framework is to introduce the topic features for

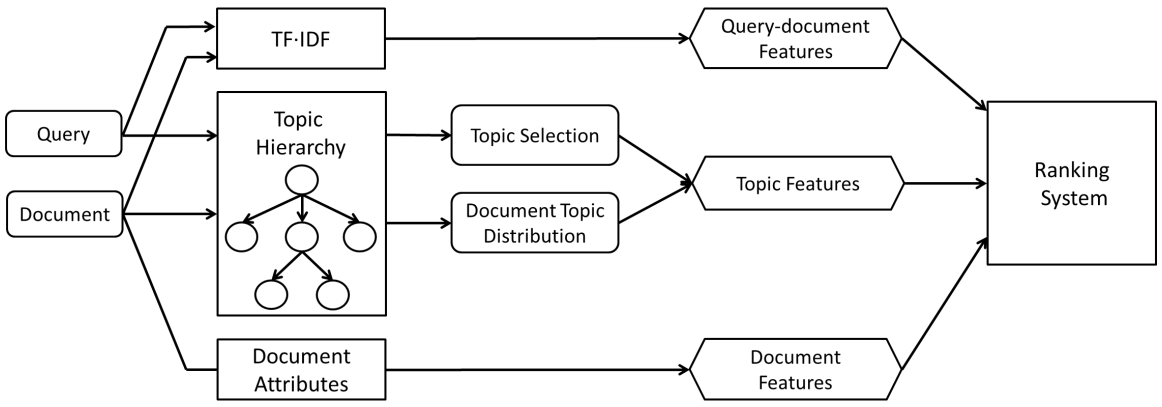


Figure 4.1: The framework for integrating hierarchical classification into the search engine with hierarchies.

a ranking system. These features are proposed to capture the topic correlation between user selected categories in the hierarchy and the indexed documents. It is computed based on the categories selected by users and the estimation of topic distribution by the integrated hierarchical classification system (the square box with caption "Topic Hierarchy" in Figure 4.1).

In the following subsections, we will firstly discuss the topic features due to their importance in our framework. Secondly, we briefly describe the other two text based features (Document Features and Query-document Features in Figure 4.1). Finally, we discuss how we integrate the three types of features into the ranking system.

### 4.2.2 Topic Features

How can we deal with the two challenges when integrating hierarchical classification into search engines with hierarchies?

The first challenge is to define a reasonable and commonly-accepted topic hierarchy. In fact, there already exist some high-quality topic taxonomies on the web, such as the Wikipedia topic hierarchy[1] and the ODP (Open Directory Project) category hierarchy[2]. Due to the large number of human editing efforts, the quality of these topic hierarchies would be much better than the topic hierarchies extracted by ad-hoc methods or tools. Thus, we can directly build the topic hierarchy by reusing these existing taxonomies.

For example, for general-purpose web search, we can use the ODP hierarchy for topic classification. ODP is a human-edited web directory where human experts organize millions of webpages into a complex hierarchy. The ODP hierarchy covers a broad scope of common topics on the web, including *Arts*, *Business*, *Computers*, *Health*, *Home*, *Recreation*, *Reference*, *Science*, *Shopping*, *Society* and *Sports*. With the high coverage of common topics, we believe that using the ODP hierarchy to classify webpages in a general web search engine is feasible. As we have already presented the details of ODP dataset and the hierarchical webpage classification algorithm in Chapter 3, we will mainly discuss the topic features derived from the hierarchical classification results in this framework.

[1]`http://en.wikipedia.org/wiki/Category:Main_topic_classifications`
[2]`http://www.dmoz.org/`

In our framework, we use two vectors to denote the user selected categories and the topic distribution of indexed documents. Consider a hierarchy $\mathcal{H}$ of $n$ categories (topics). We use a vector $T = (T_1, T_2, T_3, \ldots, T_n)$ to denote the selected topics by users where $T_i = 1$ means the topic $i$ is selected while $T_i = 0$ means it is not. In addition, based on the hierarchical "IS-A" relations, for each ancestor topic $k$ of the selected topic $i$, we can also set $T_k = 1$. For the indexed documents, we receive a document topic distribution $P_d$ generated from the hierarchical classification system. The topic distribution is a vector of $n$ probability values for all the topics in the hierarchy. It should be noted that the classification of indexed documents is performed off-line, and the results (topic probabilities) are directly indexed into search engines for fast online computation of topic features.

Now, we derive the topic features based on the topic selection $T$ and the document topic distribution $P_d$. A simple idea is to only use the topic probability at the most specific topic selected by users. However, we argue that such a simple method ignores the important hierarchical information. In our framework, it could be more useful to define the topic features for each level of the hierarchy,

$$F^{(j)}_{\text{topic}}(T, P_d) = \sum_{i \in \mathcal{H}_j} T(i) P'_d(i), \text{ where } P'_d(i) = P_d(i) \times \prod_{k \in an(i)} P_d(k) \qquad (4.1)$$

where $j$ is the jth level of the hierarchy; $\mathcal{H}_j$ is the set of topics at the level $j$; $an(i)$ is the set of all the ancestor topics of $i$; $P'_d(i)$ is the adjusted probability for topic $i$ by multiplying the probabilities of all its ancestor topics. We can see that this level-based topic feature is not only based on the topic probability at the selected specific category but also the probabilities of all the ancestor topics. Thus the hierarchical information is not lost.

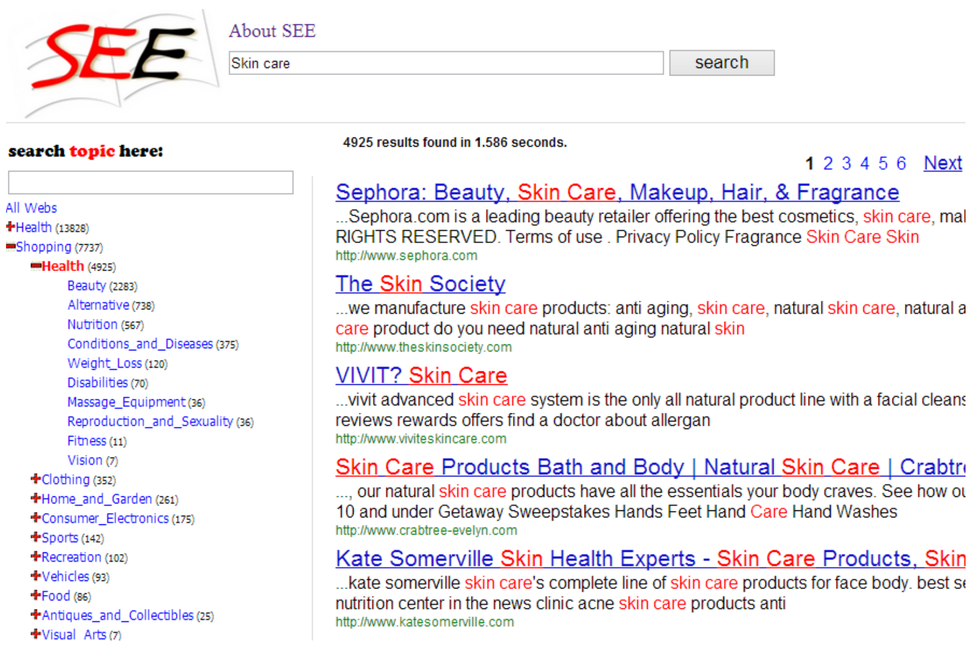
We use an example to show how our topic features are derived in a real-world search engine. Figure 4.2 shows an example of searching for "skin care" under the category hierarchy of "All" → "Shopping" → "Health" in the ODP hierarchy. In this case, the topic selection by the user is $T = (T(\text{Shopping}) = 1, T(\text{Health}) = 1)$. For an indexed document $d$, we compute the topic features for each level of the hierarchy as

1. 1st level.
$$F^{(1)}_{\text{topic}}(T, P_d) = T(\text{Shopping}) \times P_d(\text{Shopping})$$

2. 2nd level.
$$F^{(2)}_{\text{topic}}(T, P_d) = T(\text{Health}) \times P_d(\text{Health}) \times P_d(\text{Shopping})$$

For the 3rd and the 4th levels, as no topics are selected, the corresponding topic features will be zero.

It should be noted that Bennett [8] also proposed to integrate the category probabilities into the ranking. However, their problem setting is quite different to ours. In their setting, they assume that no explicit hierarchies exist for users. They have to use the query classification approach [13, 105] to estimate the categories for queries. Due to the difficulty of accurate query classification, the ranking improvement of their approach may not be stable [8]. On the other hand, in our approach, this challenge is relieved as users can directly choose appropriate categories in the hierarchies.



About SEE

Skin care | search

search topic here:

All Webs
- Health (13828)
- Shopping (7737)
  - Health (4925)
    - Beauty (2283)
    - Alternative (738)
    - Nutrition (567)
    - Conditions_and_Diseases (375)
    - Weight_Loss (120)
    - Disabilities (70)
    - Massage_Equipment (36)
    - Reproduction_and_Sexuality (36)
    - Fitness (11)
    - Vision (7)
  - Clothing (352)
  - Home_and_Garden (261)
  - Consumer_Electronics (175)
  - Sports (142)
  - Recreation (102)
  - Vehicles (93)
  - Food (86)
  - Antiques_and_Collectibles (25)
  - Visual  Arts (7)

4925 results found in 1.586 seconds.

1 2 3 4 5 6 Next

Sephora: Beauty, Skin Care, Makeup, Hair, & Fragrance
...Sephora.com is a leading beauty retailer offering the best cosmetics, skin care, mal
RIGHTS RESERVED. Terms of use . Privacy Policy Fragrance Skin Care Skin
http://www.sephora.com

The Skin Society
...we manufacture skin care products: anti aging, skin care, natural skin care, natural a
care product do you need natural anti aging natural skin
http://www.theskinsociety.com

VIVIT? Skin Care
...vivit advanced skin care system is the only all natural product line with a facial clean
reviews rewards offers find a doctor about allergan
http://www.viviteskincare.com

Skin Care Products Bath and Body | Natural Skin Care | Crabtr
..., our natural skin care products have all the essentials your body craves. See how ou
10 and under Getaway Sweepstakes Hands Feet Hand Care Hand Washes
http://www.crabtree-evelyn.com

Kate Somerville Skin Health Experts - Skin Care Products, Skin
...kate somerville skin care's complete line of skin care products for face body. best s
nutrition center in the news clinic acne skin care products anti
http://www.katesomerville.com

Figure 4.2: Searching for "skin care" under the topic hierarchy "Shopping"→"Health" in the ODP hierarchy. This is performed in a small demo of the SEEU search engine called SEE.

### 4.2.3 Text-based Features

The other two types of text-based features are document features and query-document features. We tabulate these features in Table 4.1. The meaning of document features can be self-explained in the feature description. For the important query-document features, we give a formal description on how we compute them.

In our framework, we consider six text fields in a webpage. They are *URL*, *title*, *description*, *keywords*, *body text* and *anchor text* (see how we extract them from webpages in Chapter 3). To simplify the discussion, we denote these fields as $f_i$ $(1 \leq i \leq 6)$. Give a query $q$ and a document $d$, we list the formulas to compute the query-document features in Table 4.1 as follows:

- 8. Number of matched words in a field

$$F_{\text{\#matched words}}(q, d^{(f_i)}) = |\{w \in q \cap d^{(f_i)}\}| \tag{4.2}$$

- 9. Ratio of matched words in a field

$$F_{\text{\%matched words}}(q, d^{(f_i)}) = \frac{|\{w \in q \cap d^{(f_i)}\}|}{|\{w \in d^{(f_i)}\}|} \tag{4.3}$$

- 10. Sum of TF scores of matched words in a field

$$F_{\text{TF}}(q, d^{(f_i)}) = \sum_{w \in q \cap d^{(f_i)}} TF(w, d^{(f_i)}) \tag{4.4}$$

Table 4.1: The text based features in our framework. The fields used to compute query-document features include URL, title, description, keywords, body text and anchor text.

| ID | Feature Type | Feature Description |
|---|---|---|
| 1 | Document features | Number of slashes in the URL |
| 2 | | Length of the URL |
| 3 | | Number of distinct words in a field |
| 4 | | Number of total words in a field |
| 5 | | Number of inlinks |
| 6 | | Number of outlinks |
| 7 | | PageRank score |
| 8 | Query-document features | Number of matched words in a field |
| 9 | | Ratio of matched words in a field |
| 10 | | Sum of TF scores of matched words in a field |
| 11 | | Sum of normalized TF scores of matched words in a field |
| 12 | | Sum of IDF scores of matched words in a field |
| 13 | | Sum of TF·IDF scores of matched words in a field |

- 11. Sum of normalized TF scores of matched words in a field

$$F_{\text{Norm TF}}(q, d^{(f_i)}) = \frac{1}{|\{w \in d^{(f_i)}\}|} \sum_{w \in q \cap d^{(f_i)}} TF(w, d^{(f_i)}) \tag{4.5}$$

- 12.Sum of IDF scores of matched words in a field

$$F_{\text{IDF}}(q, d^{(f_i)}) = \sum_{w \in q \cap d^{(f_i)}} IDF(w, D^{(f_i)}) \tag{4.6}$$

- 13. Sum of TF·IDF scores of matched words in a field

$$F_{\text{TF}\cdot\text{IDF}}(q, d^{(f_i)}) = \sum_{w \in q \cap d^{(f_i)}} TF(w, d^{(f_i)}) \cdot IDF(w, D^{(f_i)}) \tag{4.7}$$

where $TF(w, d^{(f_i)})$ is the term frequency of the matched term $w$ in the document field $d^{(f_i)}$; $IDF(w, D^{(f_i)})$ is the reverse document frequency of term $w$ in all the documents with text feature $f_i$.

To help readers understand how these features are actually calculated, we show the computation of the sum of TF·IDF scores as an example. Consider a query "intelligent system" and a document with title "intelligent system and technology". After we remove stop words and apply stemming in both the query and the document, the sum of TF·IDF feature for its *title* field is computed as:

$$\begin{aligned} F_{\text{TF}\cdot\text{IDF}}(q, d^{(title)}) &= TF(intellig, d^{(title)}) \times IDF(intellig, D^{(title)}) \\ &\quad + TF(system, d^{(title)}) \times IDF(system, D^{(title)}) \end{aligned}$$

### 4.2.4 Ranking Strategy

In this subsection, we present how we integrate the three types of features into a ranking system. The three types of features are all important for the final ranking. It is reasonable that the final ranking score is defined as a linear function of those features:

$$\mathbf{S}_{\mathbf{ERIC}}(\mathbf{q},\mathbf{d}) = \overrightarrow{\mathbf{W_D}} \cdot \overrightarrow{\mathbf{F_D(d)}} + \overrightarrow{\mathbf{W_{<Q,D>}}} \cdot \overrightarrow{\mathbf{F_{<Q,D>}(q,d)}} + \overrightarrow{\mathbf{W_T}} \cdot \overrightarrow{\mathbf{F_T(q,d)}} \tag{4.8}$$

where $\overrightarrow{\mathbf{F_D(d)}}$ is the vector of document-only features; $\overrightarrow{\mathbf{F_{<Q,D>}(q,d)}}$ is the vector of query-document features for all the six text fields; and $\overrightarrow{\mathbf{F_T(q,d)}}$ is the vector of topic features. Parameters $\overrightarrow{\mathbf{W_D}}$, $\overrightarrow{\mathbf{W_{<Q,D>}}}$ and $\overrightarrow{\mathbf{W_T}}$ are the weights for each feature respectively.[3]

We can see that the traditional ranking methods, such as the web directory and the keywords-based search, can be derived from the Equation 4.8. Firstly, if users simply browse a topic category without any keywords (i.e., $\mathbf{q} = \emptyset$), $\overrightarrow{\mathbf{F_{<Q,D>}(q,d)}}$ will be constant for all webpages. Our ranking system will return the most popular (large $\overrightarrow{\mathbf{F_D(d)}}$ score) and highly category-relevant (large $\overrightarrow{\mathbf{F_T(q,d)}}$ score) results at the top. In this case, our ranking system becomes a web directory, and people can browse the most popular webpages in each topic of the whole hierarchy. Secondly, if users do not choose any category (i.e., $\mathbf{T} = \emptyset$), $\overrightarrow{\mathbf{F_T(q,d)}}$ will become constant for all webpages. Our ranking system will rank all webpages only based on the document features ($\overrightarrow{\mathbf{F_D(d)}}$) and query matching features ($\overrightarrow{\mathbf{F_{<Q,D>}(q,d)}}$). In this case, our ranking system acts as the same as the flat search method. Users can just search the desired documents by keywords.

We use RankSVM [53], a popular learning to rank algorithm, to learn the weights of our ranking function. A detailed review of learning to rank algorithms can be found in Chapter 2.3. RankSVM learns the ranking model by minimizing the pairwise loss (i.e., a relevant document is ranked lower than an irrelevant document) [53, 62]. The recently proposed $SVM^{rank}$ [63] package can learn RankSVM efficiently on very large datasets.

In this section, we have presented how we integrate the probabilities of each webpage predicted by hierarchical classifiers into our novel ranking system. In the next section, we will report the evaluation results of our new ranking method compared with the traditional flat ranking methods.

## 4.3 Evaluation of Hierarchical Classification Enhanced Ranking

In this section, we conduct experiments on the well-known TREC (Text REtrieval Conference) datasets to compare our ranking algorithm with traditional flat ranking methods.

[3]In our previous paper [67], we propose a simple product based ranking function, i.e., $S(q,d,c) = F_Q(q,d) \times F_D(d) \times F_C(c,d)$. In fact, we can transform this product equation into our framework by taking logarithm on both sides.

### 4.3.1 Experimental Dataset

We use TREC[4] web search datasets to evaluate our ranking method. TREC consists of a series of workshops focusing on different information retrieval research tracks. In our experiments, we choose the four latest web track datasets in our experiments. They are from TREC 2009, 2010, 2011 and 2012 web tracks.[5] Their statistical information is shown in Table 4.2.

Table 4.2: The statistical information of the TREC datasets.

| Dataset | Documents | Unique Words |
|---|---|---|
| TREC 2009 | 23,601 | 205,135 |
| TREC 2010 | 25,329 | 218,068 |
| TREC 2011 | 19,381 | 164,516 |
| TREC 2012 | 16,055 | 124,950 |

For each TREC web track dataset, TREC offers a list of 50 ad-hoc search tasks with relevance judgement of documents related to each task. Specifically, each task consists of a description of the intended search topic and a suggested query (keywords) (See a sample in Table 4.3). The relevance judgement of each document is in a five-point scale as {-2, 0, 1, 2, 3} where -2 means spams and 3 means highly relevant results. For each task, the participants will run their own retrieval systems against the query, and submit a list of the top-ranked documents for evaluation.

Table 4.3: A sample of ad-hoc search tasks from the TREC 2010 web track.

| No | Category | Phrases |
|---|---|---|
| 1 | Find information about horse hooves, their care, and diseases of hooves. | horse hooves |
| 2 | Find events sponsored by the Association of Volleyball Professionals. | avp |
| 3 | Find locations and information of Discovery Channel stores and their products. | discovery channel store |
| 4 | Find information about iron as an essential nutrient. | iron |
| 5 | Find information about jobs in Connecticut. | ct jobs |
| 6 | Find information about penguins. | penguins |
| 7 | Find information about computer worms, viruses, and spyware. | worm |
| 8 | Find information about Flushing, a neighborhood in New York City. | flushing |
| 9 | Find information about PVC pipes and fittings. | pvc |
| 10 | Find beginners instructions to sewing, both by hand and by machine. | sewing instructions |

Given a query $q$ and a document $d$ in one of the TREC datasets, we need to calculate the

[4]The home page of TREC is `http://trec.nist.gov/`.

[5]We can only download the ground truth relevance judgement from TREC at `http://trec.nist.gov/data/webmain.html`. The actual webpages are provided by Cluweb09 project. We use *Indri* search API to download them. *Indri* can be visited at `http://boston.lti.cs.cmu.edu/Services/clueweb09_catb/` (authentication required).

features (see Section 4.2) for our ranking framework. There are three types of features we need to calculate.

- Query-document features. We can easily calculate them based on the formulas we list in Section 4.2.3.
- Document features. We can compute most of them based on the feature extraction tool (see Chapter 3.1.1). For the PageRank score, we will directly use the normalized PageRank score provided by TREC.[6]
- Topic features. We want to evaluate the effect of hierarchical classification in searching webpages. Before computing the topic features, we need to find a proper topic for each search task (or query). In our experiments, we recruit several graduate students to manually categorize all the search tasks into the ODP hierarchy (see Chapter 3.3.1) in our ranking system. For example, for the task 2 ("avp") in TREC 2010 (see Table 4.3), one possible categorization in the ODP hierarchy is "Sports". A sample of the assigned topics by students for the search tasks from the TREC 2010 dataset can be seen in Table 4.4. To get the topic features for each document, we use our hierarchical SVM classifiers (see Chapter 3.2) to predict its probability estimation, and calculate the topic features $\overrightarrow{\mathbf{F_T}(\mathbf{q}, \mathbf{d})}$ by equation 4.1.

Table 4.4: A sample of assigned topics by students for the search tasks in TREC 2010 web track dataset.

| Task | Description | Assigned Topics |
|---|---|---|
| 1 | Find information about horse hooves, their care, and diseases of hooves. | Health |
| 2 | Find events sponsored by the Association of Volleyball Professionals. | Sports |
| 3 | Find locations and information of Discovery Channel stores and their products. | Shopping |
| 4 | Find information about iron as an essential nutrient. | Health |
| 5 | Find information about jobs in Connecticut. | Business |
| 6 | Find information about penguins. | Recreation/Pets |
| 7 | Find information about computer worms, viruses, and spyware. | Computers→Security |
| 8 | Find information about Flushing, a neighborhood in New York City. | Society |
| 9 | Find information about PVC pipes and fittings. | Business→Industrial Goods and Services |
| 10 | Find beginners instructions to sewing, both by hand and by machine. | Arts→Crafts→Needlework |

[6]The PageRank scores can be freely downloaded from `http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank`.

Finally, we conduct feature scaling on each computed feature by min-max normalization in the range [0,1]:

$$S'_i(d) = \frac{S_i(d) - \min_d S_i(d)}{\max_d S_i(d) - \min_d S_i(d)} \,. \tag{4.9}$$

### 4.3.2 Evaluation Metric

To evaluate the performance of the ranking methods, we use a popular information retrieval measure, named DCG (Discounted Cumulative Gain) [59]. DCG is particularly useful to evaluate the performance when the documents are judged by graded relevance (e.g., the five-point scales in TREC) rather than binary relevance. The intuition of DCG is that the usefulness (gain) of a document in a ranking list is discounted by its position. In fact, DCG accumulates the discounted gain over all the documents up to a position. Specifically, the DCG score at a position $p$ can be defined as

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(1 + i)} \tag{4.10}$$

where $rel_i \in \{-2, 0, 1, 2, 3\}$ is the graded relevance of the result at position $i$. Usually, the DCG scores for different queries vary dramatically. To make the scores between different queries comparable, we actually use NDCG (Normalized Discounted Cumulative Gain) in our experiments:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{4.11}$$

where $IDCG_p$ is the ideal DCG score at position $p$. Thus, the DCG score is rescaled in a comparable range so that we can compare our evaluation metric between different queries.

### 4.3.3 Experiment Configuration

We compare our ranking method with the traditional flat ranking methods without hierarchy. To simplify the notation, we denote the two ranking methods as ERIC and FLAT (flat search method). For ERIC, we train $SVM^{rank}$ on all the features (in ranking Equation 4.8) including the topic features. For FLAT, we use the classic Okapi BM25 [94] ranking function. It only uses the query-document features and PageRank score. In our experiment, the Okapi BM25 ranking score is computed as

$$\mathbf{S_{BM25}(q, d)} = \left( \sum_{1 \le i \le 6} BM25(q, d^{(f_i)}) \right) \cdot F_{PageRank}(d) \tag{4.12}$$

where $BM25(q, d^{(f_i)})$ is defined as

$$BM25(q, d^{(f_i)}) = \sum_{w \in q \cap d^{(f_i)}} IDF(w, D^{(f_i)}) \cdot \frac{TF(w, d^{(f_i)}) \cdot (k_1 + 1)}{TF(w, d^{(f_i)}) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \tag{4.13}$$

where the definition of $TF(w, d^{(f_i)})$ and $IDF(w, D^{(f_i)})$ is the same as computing the query-document features in ERIC; $|d|$ is the length of the document $d$ in terms; $avgdl$ is the average document length in the dataset; $b$ and $k_1$ are the parameters.

We use a variation of the cross validation method in our experiments. When we conduct experiments for a dataset (such as TREC 2009), we choose that dataset as the testing set and use the other three datasets (i.e., TREC 2010, 2011 and 2012) as the training set. We train the ranking methods of ERIC and FLAT on the training set to tune the parameters, and evaluate their performance on the testing set respectively. We report the average NDCG scores at different positions in the set $\{3, 5, 10, 30, 50\}$. Furthermore, we measure the significance of performance difference by paired t-test at 5% significance level between ERIC and FLAT.[7] When ERIC achieves significantly better performance than FLAT, we will mark its NDCG scores in bold.

### 4.3.4 Results

#### Comparing ERIC and FLAT

Table 4.5 compares the performance of ERIC and FLAT, in terms of the NDCG scores at the positions of 3, 5, 10, 30 and 50. We can see that the results of ERIC are quite good. In all cases, ERIC always performs significantly better than FLAT. The largest performance gain (about 0.2 NDCG improvement) can be observed on TREC 2010 dataset. The performance on the other datasets also increases by 0.05 to 0.2. It clearly demonstrates that our ranking method is much better than the traditional flat ranking approach.

Table 4.5: The NDCG scores of ERIC and FLAT.

| | TREC 2009 | | TREC 2010 | | TREC 2011 | | TREC 2012 | |
|---|---|---|---|---|---|---|---|---|
| NDCG | ERIC | FLAT | ERIC | FLAT | ERIC | FLAT | ERIC | FLAT |
| N@3 | **0.307** | 0.145 | **0.275** | 0.046 | **0.251** | 0.080 | **0.189** | 0.104 |
| N@5 | **0.317** | 0.165 | **0.278** | 0.063 | **0.253** | 0.087 | **0.195** | 0.086 |
| N@10 | **0.315** | 0.183 | **0.292** | 0.080 | **0.250** | 0.094 | **0.200** | 0.095 |
| N@30 | **0.312** | 0.200 | **0.344** | 0.101 | **0.302** | 0.106 | **0.255** | 0.114 |
| N@50 | **0.344** | 0.211 | **0.369** | 0.122 | **0.333** | 0.116 | **0.290** | 0.138 |

**Discussion** Why does ERIC perform better than FLAT? We will use the task 2 (see Table 4.3) in TREC 2010 as a case study. The topic of task 2 is to find events sponsored by the Association of Volleyball Professionals (AVP) and the suggested query is “avp”. The topic of this task is about “Sports”. Figure 4.3 shows the top ten ranked results by ERIC and FLAT. We can see that by integrating the probabilities of “Sports” of each webpage, ERIC successfully returns the highly relevant results related to the Association of Volleyball Professionals. However, for FLAT, we can see that the top six ranked results are irrelevant. Actually, these results are about the movie “Alien vs. Predator” which also has an abbreviated form as “avp”. As “avp” is an ambiguous keyword in this task, FLAT without hierarchical classification can not distinguish the webpages belonging to “Sports”. Thus, its results are worse than ERIC.

[7]We use the significance test script from LETOR dataset. It can be downloaded at `http://research.microsoft.com/en-us/um/beijing/projects/letor/`.

1. **avp - about the avp**
   `http://www.avp.com/About-Us/About-the-AVP.aspx`
2. **volleyball @ avp.com: about: about the avp**
   `http://www.avp.com/about/`
3. **avp volleyball tickets - avp volleyballtickets at stubhub!**
   `http://www.stubhub.com/avp-volleyball-tickets/`
4. **avp manhattan beach open, pubclub.com!**
   `http://www.pubclub.com/specialevents/avp.htm`
5. **volleyball @ avp.com: tour schedule: 2008 avp crocs tour schedule**
   `http://web.avp.com/schedule/`
6. **volleyball @ avp.com: news: avp.com**
   `http://web.avp.com/news/`
7. **volleyball @ avp.com: photos: avp.com**
   `http://web.avp.com/photos/`
8. **volleyball @ avp.com: photos: avp.com**
   `http://www.avp.com/photos/`
9. **volleyball @ avp.com: tour schedule: avp hot winter nights tour**
   `http://web.avp.com/schedule/winter_tour.jsp`
10. **volleyball @ avp.com: avpnext: avpnext**
   `http://web.avp.com/volleyball_nation/avpnext/`

(a) Ranked results by ERIC

1. avp home — ugos world of alien vs. predator — ugo.com
   `http://avp.ugo.com/`
2. aliens vs. predator extinction - planet avp
   `http://www.planetavp.com/avpextinction/`
3. planet avp - news, screenshots, previews, reviews, guides
   `http://www.planetavp.com/`
4. avp - planet avp
   `http://www.planetavp.com/avp/`
5. planet avp: avp movie
   `http://www.planetavp.com/avpmovie/`
6. planet avp: avp movie: files
   `http://www.planetavp.com/avpmovie/files/`
7. **volleyball @ avp.com: tour schedule: 2008 avp crocs tour schedule**
   `http://www.avp.com/schedule/`
8. **volleyball @ avp.com: home: the #1 volleyball destination online!**
   `http://web.avp.com/index.jsp`
9. **volleyball @ avp.com: tour schedule: avp hot winter nights tour**
   `http://web.avp.com/schedule/winter_tour.jsp`
10. planet avp: avp requiem
   `http://www.planetavp.com/avp2movie/`

(b) Ranked results by FLAT

Figure 4.3: Ranked results for query "avp" in TREC 2010 by ERIC and FLAT. The relevant results are marked in bold.

### Comparing ERIC and FLAT with category keywords

It may be argued that as we already use the category information (as hierarchical probabilities output by SVMs) in ERIC, it could be unfair to only use the keywords and webpage importance for FLAT. Thus, in our next experiment, for FLAT, we add the category phrase into the query keywords and repeat the experiments for FLAT on all the datasets. For example, when conducting experiments for the "avp" task in FLAT, we change its query to "avp sports". We compare the results between ERIC and FLAT with category keywords. The results are shown in Table 4.6. Comparing Table 4.6 and Table 4.5, we find that in most cases adding the category phrase into the query keywords for FLAT does improve its performance. However, compared to the new results of FLAT, ERIC still consistently performs better in all cases.

Table 4.6: The NDCG score of ERIC and FLAT with category keywords.

| | TREC 2009 | | TREC 2010 | | TREC 2011 | | TREC 2012 | |
|---|---|---|---|---|---|---|---|---|
| NDCG | ERIC | FLAT | ERIC | FLAT | ERIC | FLAT | ERIC | FLAT |
| N@3 | **0.307** | 0.188 | **0.281** | 0.128 | **0.256** | 0.109 | **0.185** | 0.096 |
| N@5 | **0.317** | 0.184 | **0.284** | 0.134 | **0.260** | 0.109 | **0.192** | 0.104 |
| N@10 | **0.315** | 0.200 | **0.298** | 0.150 | **0.257** | 0.120 | **0.197** | 0.122 |
| N@30 | **0.312** | 0.213 | **0.348** | 0.183 | **0.307** | 0.080 | **0.253** | 0.135 |
| N@50 | **0.344** | 0.221 | **0.375** | 0.210 | **0.337** | 0.054 | **0.288** | 0.149 |

**Discussion** Why does FLAT with category keywords still perform worse than ERIC? Here, we still use the "avp" example for discussion. Figure 4.4 shows the results of FLAT with category keywords. We can see that by combining the suggested query with the category phrase (i.e., "avp Sports"), the returned results are still not good. Firstly, there are still two pages (i.e., the 4th and the 10th pages) related to the movie "Alien vs. Predator". This is because the word "Sports" also appears in the two pages. Secondly, the top three ranked results are still not relevant. Although both words appear in these pages, they are more likely to be relevant to the general word "Sports" rather than the Association of Volleyball Professionals ("avp" ). Moreover, we also find that the relevant results (i.e., 7th, 8th and 9th results) in Figure 4.3(b) are filtered out. This is because the word "Sports" does not appear in these pages. This experiment confirms that simple keyword queries are often insufficient to express topics as keywords [29].

1. buy sports tickets online from a sports ticket broker, agency for philadelphia, chicago, new york, los angeles
   `http://www.abctickets.com/sports/`
2. indiana university
   `http://www.indiana.edu/`
3. tickets at stubhub! where fans buy and sell tickets
   `http://www.stubhub.com/`
4. avp home — ugos world of alien vs. predator — ugo.com
   `http://avp.ugo.com/`
5. **huntington beach events calendar information**
   `http://www.huntingtonbeachevents.com/`
6. board of directors - associated content
   `http://www.associatedcontent.com/company_directors.shtml`
7. gamespy: game sites
   `http://www.gamespy.com/network/`
8. **avp manhattan beach open, pubclub.com!**
   `http://www.pubclub.com/specialevents/avp.htm`
9. **volleyball @ avp.com: about: avp media guide**
   `http://www.avp.com/about/mediaguide.jsp`
10. jammsbro s movie news: official avp2 website up and running
   `http://alelbert.xm.com/topic-1306.htm`

Figure 4.4: Ranked results for query "avp Sports" in TREC 2010 by FLAT. The relevant results are marked in bold.

### 4.3.5 Analysis of Ranking Feature Importance

In this subsection, we study the importance of different features of the learned ranking models. Recalling that our ranking model is in the form of a linear regression function (see Equation 4.8), we analyze the weights of the linear regression function. In this analysis, we report the top ten most important features of the ranking model trained for the TREC 2010 dataset (see Figure 4.5). The results on other datasets are similar.

Firstly, we analyze the query-document features. From Figure 4.5(a), we can see that nine out of the top ten features (marked by bullets) are related to the meta text of webpages (i.e., title, description, keywords, anchor text); only one feature is related to the body text. This result shows that matching keywords in the meta text of webpages for ranking is very important.

Secondly, we analyze the document features from Figure 4.5(b). We can see that the weights of both inlinks and outlinks are positive. It means that webpages with rich link connection may be more relevant. It is interesting to see that the URL related features (length and number of slashes) have small or even negative weights. It means that our ranking model may give high relevance scores to webpages with short URLs. This is reasonable because webpages with short URLs are usually website portals which are more important. Surprisingly, we find that the sign of PageRank weight is negative. This is somehow contrast to the conclusion from [85]. To analyze this issue, we plot the histogram of the PageRank scores for both the relevant

| Field | Feature | Weight |
|---|---|---|
| • Description | Sum of TF score of matched words | 16.129 |
| • Anchor Text | Sum of TF·IDF scores of matched words | 7.583 |
| • Keywords | Sum of TF score of matched words | 5.932 |
| • Title | Number of total words | 4.756 |
| Body | Number of matched words | 4.487 |
| • Anchor Text | Number of total words | 3.052 |
| • Title | Sum of IDF scores of matched words | 3.032 |
| • Keywords | Number of total words | 2.702 |
| • Description | Number of matched words | 2.592 |
| • Anchor Text | Number of matched words | 1.720 |

(a) Top ten ranked query-document features. Meta text features are marked by bullets.

| Feature | Weight |
|---|---|
| Number of outlinks | 2.393 |
| Number of inlinks | 1.765 |
| Length of the URL | 0.424 |
| Number of slashes in the URL | -0.302 |
| PageRank score | -0.414 |

(b) Ranked document features

| Feature | Weight |
|---|---|
| 1st level topic feature | 1.408 |
| 2nd level topic feature | 0.675 |
| 3rd level topic feature | 0.163 |

(c) Ranked topic features

Figure 4.5: Most important ranking features with learned weights from the ranking model trained on the TREC 2010 dataset.

and the irrelevant webpages in the training set of TREC 2010 dataset[8] (see Figure 4.6). We can see that for most of the PageRank scores, the irrelevant results occur far more often than the relevant results. This observation is consistent with the learned weight of PageRank feature that penalizes webpages with large PageRank scores.

Thirdly, we analyze our proposed topic features. We can see from Figure 4.5(c) that all the topic feature weights are positive. It means that they all have a positive contribution to the final relevance scores. Although the absolute weight value of the topic features are smaller than the other two types of features. It does not mean that the topic features are not important. For example, Figure 4.7 shows the contribution of the three types of features for the ranking of "avp" example in TREC 2010 dataset. We can see a clear pattern that for the relevant results (with relevance scores 1, 2, or 3), topic features have more than 40% contribution, while for the irrelevant results (with relevance scores -2 or 0), the corresponding contribution by topic

[8]The training examples of TREC 2010 dataset come from TREC 2009, 2011 and 2013 datasets.

features is less than 5%. This analysis clearly demonstrates that our proposed topic features are quite effective for boosting the ranking of relevant results above irrelevant results in the search engine with hierarchies.

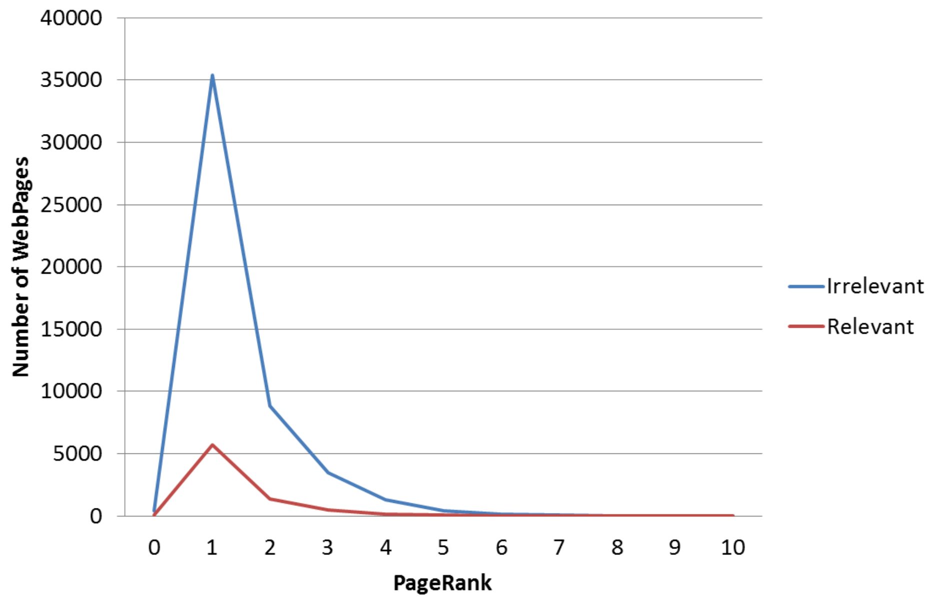


Figure 4.6: PageRank histogram for the irrelevant and relevant webpages on TREC 2010 dataset. The X axis are the PageRank scores.

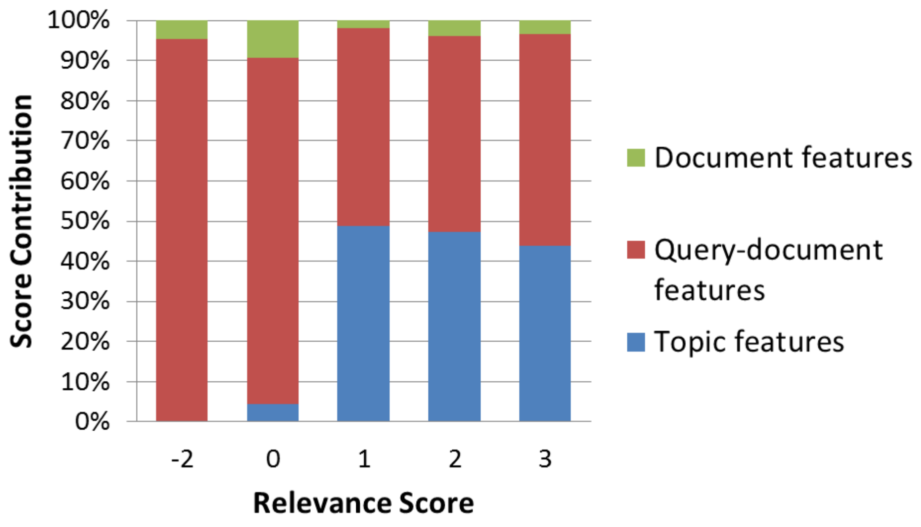


Figure 4.7: The score contribution for the “avp” example in TREC 2010 dataset.

To summarize, in this section, we conduct experiments on the TREC datasets to evaluate our proposed ranking framework ERIC. From the experimental results, we find that by seamlessly integrating hierarchical text classification into ranking, our method can significantly outperform the traditional flat search methods.

## 4.4 Summary

In this chapter, we present a novel ranking framework, called ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), for search engines with hierarchies, and report our experimental results on the well-known TREC (Text REtrieval Conference) web search datasets. From the experimental results, we find that by seamlessly integrating hierarchical text classification and learning to rank methods into the search engine, our framework with hierarchical classification can outperform the traditional flat search methods significantly.

# Chapter 5

# Improving Hierarchical Classification by Active Learning

Hierarchical classification is important for building search engines with hierarchies. To build an accurate hierarchical classification system with many categories, usually a very large number of documents must be labeled and provided. This can be very costly. Active learning has been shown to effectively reduce the labeling effort in traditional (flat) text classification, but few work have been done in hierarchical text classification due to several challenges. A major challenge is to reduce the so-called *out-of-domain* (defined later) queries. Previous state-of-the-art approaches tackle this challenge by simultaneously forming the unlabeled pools on all the categories regardless of the inherited hierarchical dependence of classifiers.

In this chapter, we propose a novel top-down hierarchical active learning framework with effective strategies to tackle this and other challenges. With extensive experiments on eight real-world hierarchical text datasets, including the RCV1-v2 and ODP datasets, we demonstrate that our strategies are highly effective, and they outperform the state-of-the-art hierarchical active learning methods by reducing 20% to 40% queries.

This work was in collaboration with Dr. Charles Ling and Dr. Huaimin Wang. We published the results in the *Proceeding of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (PAKDD 2013) [76].

## 5.1 Introduction

Given documents organized in a meaningful hierarchy (such as a topic hierarchy), it is much easy for users to browse and search the desired documents. Thus, hierarchical text classification is an important task in many real-world applications, which include, for example, news article classification [71], webpage topic classification [38, 22, 81] and patent classification [40].

In hierarchical text classification, a document is assigned with multiple suitable categories from a predefined hierarchical category space. Different from traditional flat text classification, the assigned categories for each document in the hierarchy have inherited hierarchical relations. For example, in the hierarchy of the Open Directory Project[1] (ODP), one path of

[1]http://www.dmoz.org/

the hierarchy (see Figure 5.1) includes *Computers* (Comp.) → *Artificial Intelligence* (A.I.) → *Machine Learning* (M.L.). Any webpage belonging to *M.L.* also belongs to *A.I.* and *Comp.*.

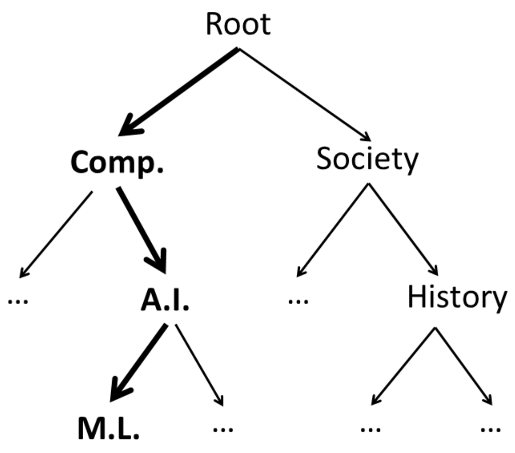


Figure 5.1: A partial hierarchy of ODP.

In this chapter, we study machine learning approaches for building a hierarchical classification system. According to [106], the most effective and appropriate approach for building a hierarchical classification system is to train a binary classifier on each category of the hierarchy (see detailed review in Chapter 2.2). To train an accurate hierarchical classification system with many categories, usually a very large number of labeled documents must be provided for a large number of classifiers. However, labeling a large number of documents in a large hierarchy is very time-consuming and costly. This severely hinders the training of accurate hierarchical classification systems.

Active learning has been studied and successfully applied to reduce the labeling cost in binary text classification [112, 96, 118]. In active learning, in particular the pool-based active learning, the learner intelligently selects the most informative unlabeled example from the unlabeled pool to query an oracle[2] for the label. This can lead to a good classification model with a much smaller number of labeled examples, compared to traditional passive learning. Several works have extended binary active learning to multi-class and multi-label text classification [15, 39, 120]. Basically, they use the one-vs-rest approach to decompose the learning problem to several binary active learning tasks.

However, active learning has not been widely studied for hierarchical text classification. The key question is how to effectively select the most useful unlabeled examples for a *large* number of *hierarchically* organized classifiers. Many technical challenges exist. For example, how should the unlabeled pool be formed for each category in the hierarchy? If not formed properly, the classifier may select many so-called *out-of-domain* examples from the pool. For example, in Figure 5.1, a classifier on *A.I.* is trained on the examples under the category of *Comp.*. These examples are called *in-domain* examples for *A.I.*. Examples not belonging to *Comp.* are the so-called *out-of-domain* examples for *A.I.*. If an unlabeled example selected by

[2]In the literature of active learning, an oracle means a human being or an artificial system that can provide labels for examples.

the classifier for *A.I.* is an *out-of-domain* example, such as a document belonging to *Society*, the oracle will always answer "no", and such a query will be virtually useless, and thus wasted in training the classifier for *A.I.*. Thus, avoiding the *out-of-domain* examples for hierarchical classifiers is very important.

As far as we know, only one work [74] has been published previously on hierarchical active learning. To solve the *out-of-domain* problem, the authors use the prediction of higher-level classifiers to refine the unlabeled pools for lower-level classifiers. In their approach, the quality of the lower-level unlabeled pools depends critically on the classification performance of the higher-level classifiers. However, the authors seemed not to pay enough attention to this important fact, and their methods allow all classifiers to simultaneously select examples to query oracles (see Section 5.2 for a review). This still leads to a large number of *out-of-domain* queries, as we will show in Section 5.4.3.

As the hierarchical classifiers are organized based on the top-down tree structure, we believe that a natural and better way to form the unlabeled pools is also in the top-down fashion. In this chapter, we propose a novel top-down active learning framework, to effectively form the unlabeled pools, and select the most informative, *in-domain* examples for the hierarchical classifiers. Under our top-down active learning framework, we discuss effective strategies to tackle various challenges encountered. With extensive experiments on eight real-world hierarchical text datasets, including the RCV1-V2 and ODP datasets, we demonstrate that our method is highly effective, and it outperforms the state-of-the-art hierarchical active learning methods including [74] by reducing the number of queries 20% to 40%.

The rest of this chapter is organized as follows. In Section 5.2, we review the state-of-the-art hierarchical active learning method. In Section 5.3, we present our top-down hierarchical active learning framework. Section 5.4 describes the experimental methodology and reports the experimental results. The last section contains the summary.

## 5.2 Previous Works

To the best of knowledge, only one work [74] has been published previously in active learning for hierarchical text classification. We call it the *parallel* active learning framework. In their approach, at each iteration of active learning (see Figure 5.2), the classifiers for all categories *independently* and *simultaneously* query the oracles for the corresponding labels. To avoid selecting the *out-of-domain* examples, they use the prediction of higher-level classifiers to refine the unlabeled pools for lower-level classifiers. Specifically, an unlabeled example will be added into the lower-level unlabeled pools only if its predictions from all the ancestor classifiers are positive.

A drawback of their approach is that they do not consider the hierarchical dependence of classification performance of the classifiers in their framework but allow all classifiers to *simultaneously* form the pools and select examples to query oracles. Consider a typical running iteration of their approach (see Figure 5.2). If the quality of the unlabeled pool $\mathcal{U}_{comp}$ (formed by the classifier for *Comp.*) is not good, possibly many *out-of-domain* examples (e.g., examples from *Society*) may still be selected by the classifiers for *A.I.*. This will lead to a large number of *out-of-domain* (wasted) queries, as we will show in Section 5.4.3.

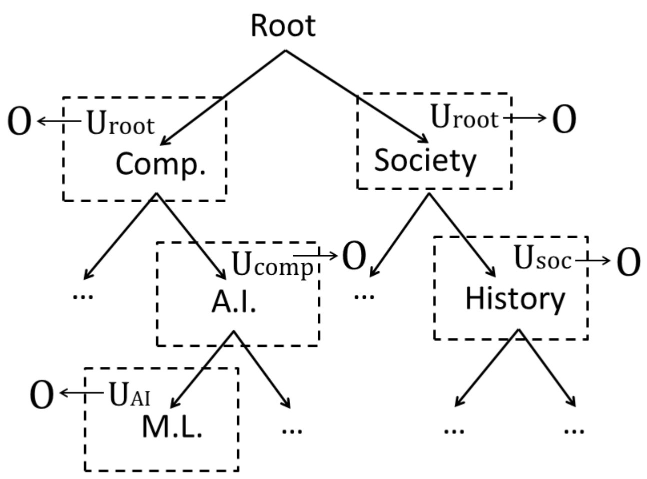


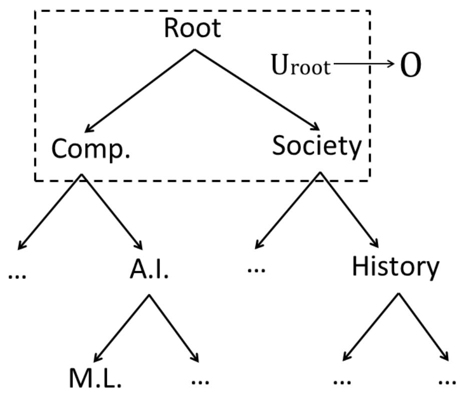
Figure 5.2: A typical iteration of the parallel active learning framework. Multiple active learning processes (represented by dashed windows) are simultaneously conducted. U denotes the unlabeled pool and O denotes the oracle. The horizontal arrows mean querying the oracle while the down arrows mean building the unlabeled pools.

How can we effectively solve the *out-of-domain* problem and the other challenges to improve active learning in hierarchical text classification? As the hierarchical classifiers are organized based on the top-down tree structure, we believe that a natural and better way to do active learning in hierarchical text classification is also in the top-down fashion. In the next section, we propose a new top-down active learning framework for hierarchical text classification to effectively tackle these challenges.

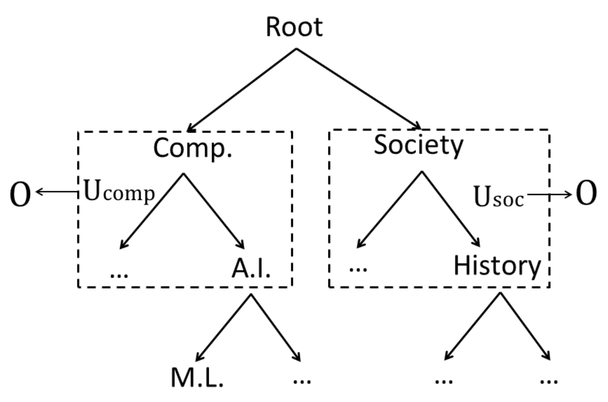
## 5.3 Top-down Hierarchical Active Learning Framework

In this section, we propose our top-down hierarchical active learning framework. Different to the parallel framework which simultaneously forms the unlabeled pools for all categories, our top-down approach forms the unlabeled pools in the top-down fashion. We use Figure 5.3 to describe our basic idea.

In Figure 5.3(a), we start active learning at the top level of the hierarchy. The top-level classifiers for *Comp.* and *Society* select examples from the global unlabeled pool $\mathcal{U}_{root}$ to query the oracle for the labels of top-level categories. The answered examples from the oracle will be used to form the unlabeled pools $\mathcal{U}_{comp}$ and $\mathcal{U}_{soc}$. After the top-level classifiers are well trained (estimated by our stopping strategy. see Section 5.3.2), we start active learning in the second level. In Figure 5.3(b), the second-level classifiers for A.I. (or History) and its sibling categories select examples from the unlabeled pool $\mathcal{U}_{comp}$ (or $\mathcal{U}_{soc}$) to query the oracle. As the examples in $\mathcal{U}_{comp}$ (or $\mathcal{U}_{soc}$) have true labels of *Comp.* (or *Society*) which are answered by the oracle, we can ensure that the second-level classifiers will not select *any out-of-domain* examples.



(a) The top level learning stage.



(b) The second level learning stage.

Figure 5.3: Examples of two typical active learning stages in the top-down active learning framework. Only partial nodes in the hierarchy are allowed to do active learning. The notations in this figure follow Figure 5.2.

Comparing Figure 5.2 and Figure 5.3, we can see that the main difference between the parallel framework and our top-down framework is which nodes are chosen to do active learning (the dashed windows in both figures) at each iteration. The parallel framework chooses all the nodes while our top-down framework only chooses a subset of appropriate nodes in the top-down fashion. We call the set of those nodes as *working set*, denoted by $\mathcal{W}$. We present the pseudo code of our top-down active learning framework:

```
Input: Query budget $B$
Output: Classifiers for all nodes
1 repeat
2     Add the root nodes $n_0$ into $\mathcal{W}$;
3     repeat
4         Select examples from $\mathcal{U}_n$ to query oracles and update children classifiers for each node
          $n$ in $\mathcal{W}$ until its stopping criteria is satisfied;
5         Form the unlabeled pools for the children nodes of the finished nodes;
6         Replace the finished nodes in $\mathcal{W}$ with their children nodes;
7     until $\mathcal{W}$ is empty;
8 until $B = 0$;
```

In our top-down active learning framework, two critical challenges need to be resolved for effective active learning. The first challenge is that the unlabeled pools may be too small. We use the examples answered by the oracle to form the unlabeled pools, and they can be too small for lower-level classifiers to learn effectively. The problem may become worse when active learning is applied to even lower-level categories. The second challenge is how do we stop learning as it is critical for the effective scheduling of active learning at different levels. We will tackle the two challenges in the following subsections.

### 5.3.1 Dual-pool Strategy

For the second and lower-level nodes, we need to form the unlabeled pools that are large enough but have few *out-of-domain* examples. In this section, we propose a novel dual-pool strategy to enlarge the unlabeled pools. Two different unlabeled pools will be built: the *answered pool* and the *predicted pool*.

**Answered pool** Our top-down active learning framework schedules the nodes to query the oracle from the top level to the bottom level. For a node (category) in the working set, we ask oracles for the labels of its children categories. For a child category $c$, among the answered examples from the oracle, the *positive* examples of $c$ will be used to form the unlabeled pool for the children categories of $c$. The *negative* examples will not be used as they are already *out-of-domain*. By doing so, we can ensure that no *out-of-domain* examples will be selected into the unlabeled pools of children categories. We call such a pool the *answered pool* and use $\mathcal{U}^a$ to denote it.

**Predicted pool** The quality of the answered pool $\mathcal{U}^a$ is perfect. However, as the size of $\mathcal{U}^a$ depends on the positive class ratio of the ancestor nodes, it could be very slow to accumulate enough examples. Thus, we can also use the prediction of the higher-level classifiers to enlarge the unlabeled pools. Although this method is also used in the parallel framework (see Section 5.2), it should be noted that when we build the lower-level unlabeled pools, the higher-level classifiers are already assumed to be well-trained. The prediction of higher-level classifiers would be accurate. Thus, the risk of introducing *out-of-domain* examples would be much smaller than the parallel framework. We call the pool built by this method as *predicted pool*, denoted by $\mathcal{U}^p$.

**Refiltering dual pools** We have two unlabeled pools for each node $n_i$ in our top-down framework. When we select a batch of examples to query the oracle, a natural question is how do we allocate the batch of queries to each pool? On one hand, the quality of the answered pool $\mathcal{U}_i^a$ is perfect but the uncertain (useful) examples may be too few due to the small pool size; on the other hand, more useful examples may exist in the larger predicted pool $\mathcal{U}_i^p$ but we may take the risk of selecting the *out-of-domain* examples. To balance the tradeoff, we propose a *refiltering strategy* for allocating the queries to both $\mathcal{U}_i^a$ and $\mathcal{U}_i^p$.

Our basic idea is to filter out the certain examples from the pools before we allocate the batch of queries. Specifically, given the batch size $M$, we firstly filter out the certain examples from both $\mathcal{U}_i^a$ and $\mathcal{U}_i^p$ to generate two small candidate pools $C_i^a$ and $C_i^p$. The filtering threshold will be empirically tuned in our experiments (See Section 5.4.3). As the examples in $C_i^a$ are all perfect (answered by oracle) and uncertain (worthy to learn), we put more queries into the perfect candidate pool $C_i^a$ by allocating $\min\{|C_i^a|, M\}$ queries. The rest of the queries will be allocated to $C_i^p$.

### 5.3.2 Stopping Strategy

An important factor of our top-down hierarchical active learning framework is knowing when to stop learning for the nodes in the working set. In other words, how do we estimate if the classifiers are well-trained or not? A heuristic approach is to estimate the classification performance by cross-validation. However, from our pilot experiments, such a method is quite unstable due to the small size of the labeled examples in active learning.

In this chapter, we adopt a simple yet effective approach to stop learning. Simply speaking, if no uncertain examples can be further selected from the candidate pools, we stop learning. This is reasonable as querying very certain examples can not improve the classification performance [126]. In our top-down framework, this strategy can be implemented by checking the size of the two candidate pools $C^a$ and $C^p$. If both pools are empty, that means all the examples in the unlabeled pools are very certain, we stop learning.[3]

To summarize, in this section, we propose our top-down hierarchical active learning framework with several strategies to tackle the *out-of-domain* problem and the other challenges encountered. In the next section, we will conduct extensive experiments to verify the effectiveness of our framework.

## 5.4 Experiments on Hierarchical Text Classification

In this section, we conduct extensive empirical studies to evaluate our top-down hierarchical active learning framework compared to the state-of-the-art hierarchical active learning approaches.

### 5.4.1 Datasets

We use eight real-world hierarchical text datasets in our experiments (see Table 5.1). The first three datasets (20 Newsgroup, OHSUMED and RCV1-V2) are common benchmark datasets for evaluation of text classification methods. The other five datasets are webpages collected from the Open Directory Project (ODP).

Table 5.1: The statistics of the datasets. Cardinality is the average categories per example (multi-label).

| Dataset | Examples | Features | Nodes | Cardinality | Height |
|---|---|---|---|---|---|
| 20 Newsgroup | 18,774 | 61,188 | 27 | 2.20 | 3 |
| OHSUMED | 16,074 | 12,427 | 86 | 1.92 | 4 |
| RCV1-V2 | 23,149 | 47,152 | 96 | 3.18 | 4 |
| Astronomy | 3,308 | 54,632 | 34 | 1.91 | 4 |
| Biology | 17,450 | 148,644 | 108 | 3.03 | 4 |
| Chemistry | 4,228 | 56,767 | 34 | 1.44 | 4 |
| Earth Sciences | 5,313 | 71,756 | 58 | 2.16 | 4 |
| Math | 11,173 | 108,559 | 107 | 1.93 | 4 |

The first dataset is *20 Newsgroups*[4], a collection of news articles partitioned evenly across 20 different newsgroups. We manually group these categories into a meaningful three-level hierarchy. The second dataset is *OHSUMED*[5], a clinically-oriented MEDLINE dataset. We use the subcategory *heart diseases* which is also used by [68, 97]. The third dataset is *RCV1-V2* [71], a news archive from Reuters. We use the 23,149 documents from the topic classification task in our experiments.[6] The other five datasets are webpages collected from the ODP. ODP

[3]For the root node which selects examples from the very large global unlabeled pool, this stopping strategy could be very slow. Thus, we empirically set 25% remained budget as the query limit for the root node.

[4]`http://people.csail.mit.edu/jrennie/20Newsgroups/`

[5]`http://ir.ohsu.edu/ohsumed/`

[6]`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

is a web directory with a complex topic hierarchy. In our experiments, we focus on a subset of the webpages extracted from the *Science* subtree.[7] The original Science subtree has more than 50 subcategories. We choose five subcategories closely related to academic disciplines.[8] They are *Astronomy*, *Biology*, *Chemistry*, *Earth Sciences* and *Math*.

For each dataset, we use bag-of-words model to represent documents. Specifically, each document is represented by a vector of term frequencies. We use Porter Stemming to stem each word and remove the rare words occurring less than three times. Small categories which have less than ten documents are also removed.

### 5.4.2 Experiment Configuration

We adopt the hierarchical SVMs [81, 110, 119] as the base learner. In each category, a linear SVM classifier is trained to distinguish its sibling categories under the same parent category. We use LIBLINEAR [41] as the implementation of linear SVM. Following the configuration of [74], we set up the penalty $C$ as 1,000 and the cost coefficient $w$ as the ratio of negative examples in the training set. Other parameters of LIBLINEAR are set to the default values.

To evaluate the performance of hierarchical classifiers, we use the hierarchical F-measure [115, 106, 74], a popular performance measure in hierarchical text classification. It is defined as,

$$hF = \frac{2 \times hP \times hR}{hP + hR}, \quad hP = \frac{\sum_i |\hat{P}_i \bigcap \hat{T}_i|}{\sum_i |\hat{P}_i|}, \text{ and } hR = \frac{\sum_i |\hat{P}_i \bigcap \hat{T}_i|}{\sum_i |\hat{T}_i|} \tag{5.1}$$

where $hP$ is the hierarchical precision and $hR$ is the hierarchical recall. $\hat{P}_i$ is the hierarchical categories predicted for test example $x_i$ while $\hat{T}_i$ is the true categories of $x_i$.

We compare our top-down framework with the parallel framework [74] and the baseline random approach. For our top-down framework, we use *average uncertainty* [39] as the informativeness measure. It measures the example based on the average uncertainty among all child classifiers under the same parent. For the parallel framework, we choose *uncertainty sampling* [112] which is also used in their experiments. For both approaches, the uncertainty of an example is measured by the absolute SVM margin score. For the random approach, we simply select the examples randomly from the global unlabeled pool.

Now, we describe the active learning setting in our experiments. We set up the total query budget as 1000. The active learning experiment is decomposed into several iterations. In each iteration, each node in the working set selects $M$ examples to query the oracle. Similar to [74], the batch size $M$ is set as the logarithm of the unlabeled pool size on each category. We use a simulated oracle in our experiments. When receiving a query, the oracle replies with the true labels for all its subcategories. It should be noted that in [74], each query only returns one label. To make a fair comparison, we also return the labels of all the subcategories for the parallel framework and the random approach. After receiving the oracle answers, we update the labeled dataset, retrain the classification models and record the F-measure results on the testing set.

[7]It can be freely downloaded at `http://olc.ijs.si/dmozReadme.html`.

[8]Most of the other subcategories are A-Z index lists and non-academic topics (e.g., Publications and Conferences).

To avoid the impact of randomness, we use 10-fold cross validation to evaluate the performance of active learning approaches. Specifically, when conducting active learning experiments on each dataset, we randomly split the dataset into 10 subsets with equal size. Of the 10 subsets, one set is retained as testing data. For the remaining nine sets, we randomly sample 0.1% data as the labeled set. The remaining examples will be used as the unlabeled pool. The active learning experiments are then repeated 10 times. The final results are averaged over the 10 runs and accompanied by the error margins with 95% confidence intervals.

### 5.4.3 Experimental Results on Benchmark Datasets

Before the experiments, we set up the parameters for our top-down framework. We need to decide a proper uncertainty threshold to filtering out certain examples (see dual-pool strategy in Section 5.3.1). As the SVM margin score based uncertainty is not comparable, we normalize it by the function $g(f) = exp(-\frac{f^2}{0.01})$ $(0 < g \leq 1)$ where $f$ is the SVM margin score. We compare the uncertainty thresholds in different values from 0.1, 0.2, to 0.9 on the RCV1-V2 dataset. The results are plotted in Figure 5.4. We find that generally the larger the threshold is, the better the performance is. Thus, we use 0.9 as the uncertainty threshold in our experiments.

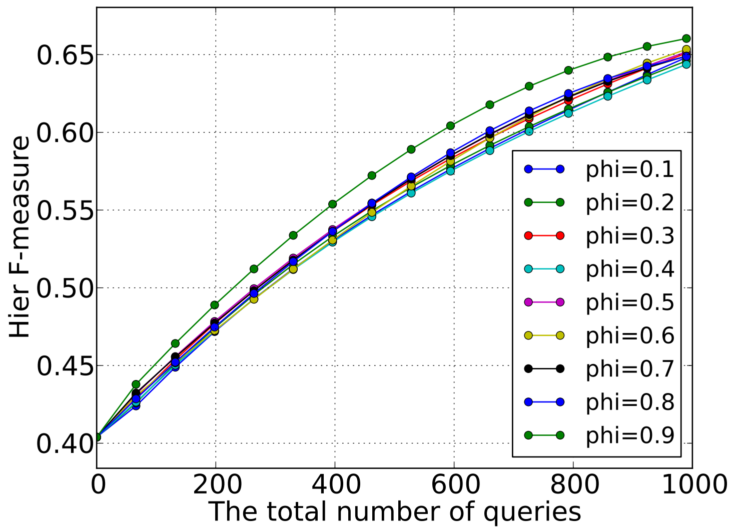
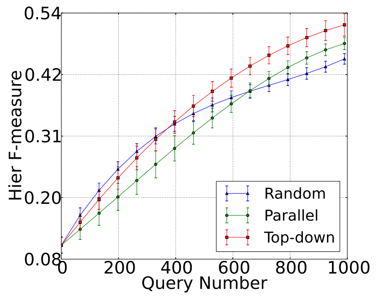
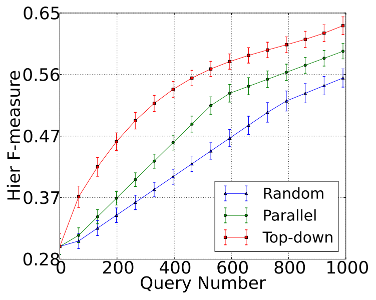


Figure 5.4: Hierarchical F-measure on RCV1-V2 dataset with different uncertainty thresholds from 0.1 to 0.9..

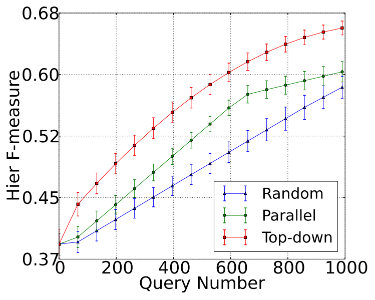
Firstly, we discuss the experimental results on the three benchmark datasets (20 Newsgroup, OHSUMED and RCV1-V2). Figure 5.5 shows the performance curves of hierarchical F-measure averaging over 10 runs. We can see that our top-down approach (framework) outperforms the parallel approach and the random approach significantly on all datasets. Specifically, on the OHSUMED and RCV1-V2 datasets, the performance curves of our top-down approach dominate the parallel approach and the random approach throughout all iterations. On the 20 Newsgroup dataset, surprisingly, during the earlier stage of active learning (before 400 queries), we observe the overlap of performance curves of our top-down approach and the random approach. The parallel approach performs even worse. This could be due to the poor initial classification performance (smaller than 0.1). However, after around 500 queries, our approach starts to outperform the random approach and the parallel approach and keeps the dominant margin till the end.



(a) 20 Newsgroup



(b) OHSUMED



(c) RCV1-V2

Figure 5.5: Hierarchical F-measure on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets.

We examine the ratio of *out-of-domain* queries. Figure 5.6 shows the average *out-of-domain* ratios on the three datasets. We can see that our top-down approach has a huge reduction of the *out-of-domain* queries. Among the three datasets, our top-down approach issues less than 10% *out-of-domain* queries. By analyzing the experiment logs of our top-down approach, we discover that for the second and the lower-level, on average about 40% queries are allocated to the answered pools (see Section 5.3.1). As the labels in the answered pools are given by the oracle, the quality of the selected examples is perfect. Thus, no *out-of-domain* examples will be selected. The observed few *out-of-domain* examples only occur in the predicted pools. The low ratio also indicates that the predicted pools built by our dual-pool strategy are much more accurate than the parallel framework. This explains why our top-down active learning approach is more effective than the parallel approach.

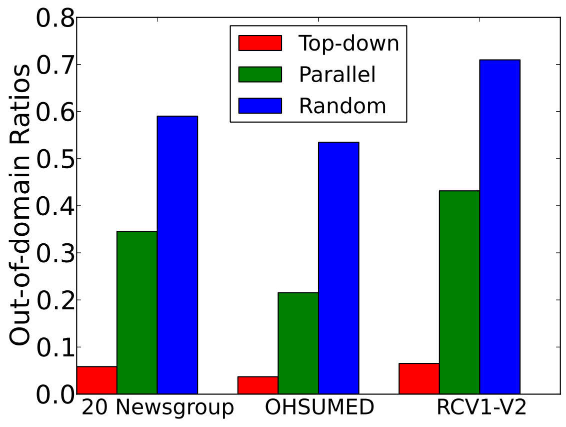


Figure 5.6: The *out-of-domain* ratios of the queries on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets.

We also study how many queries can be saved by our top-down approach. For the three approaches, we record their best performance and the queries needed in Table 5.2. We find that to achieve the best performance of the parallel approach, our top-down approach needs much fewer queries. About 20% to 37% queries can be saved. For example, on the RCV1-V2 dataset, the parallel approach needs 1,000 queries to achieve 0.606 hierarchical F-measure, while our top-down approach only requires 630 queries. Thus, (1000 – 630)/1000 = 37% queries are saved. Compared to the random approach, the query reduction is even more significant (about 30% to 56%). It clearly indicates that our top-down approach is more effective in reducing the queries than the parallel approach and the baseline random approach.

Table 5.2: The best hierarchical F-measure with needed queries on the 20 Newsgroup, OHSUMED and RCV1-V2 datasets. The value in the bracket is the relative query reduction.

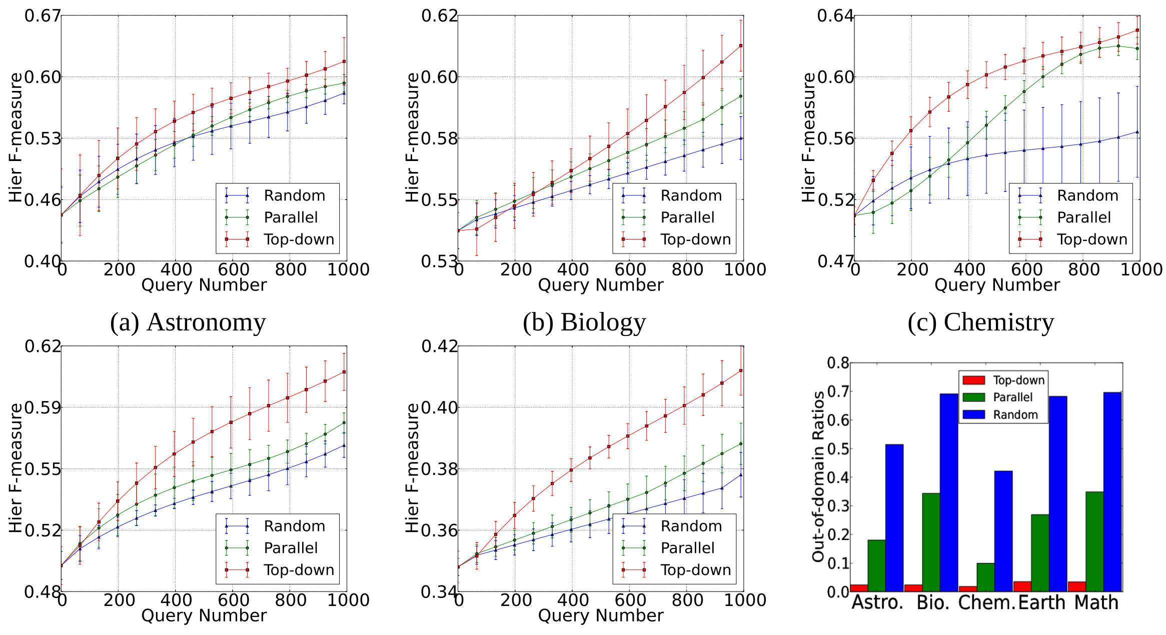
| | Method | Hier F1 | Random | Parallel | Top-down |
|---|---|---|---|---|---|
| 20 Newsgroup | Random | 0.455 | 1000 | 850 (15%) | 700 (30%) |
| | Parallel | 0.483 | | 1000 | 800 (20%) |
| | Top-down | 0.518 | | | 1000 |
| OHSUMED | Random | 0.552 | 1000 | 720 (28%) | 440 (56%) |
| | Parallel | 0.591 | | 1000 | 680 (33%) |
| | Top-down | 0.630 | | | 1000 |
| RCV1-V2 | Random | 0.587 | 1000 | 660 (34%) | 490 (51%) |
| | Parallel | 0.606 | | 1000 | 630 (37%) |
| | Top-down | 0.661 | | | 1000 |

### 5.4.4 Experimental Results on ODP datasets

In the following experiments, we compare the performance of the three approaches on five ODP datasets. From Figure 5.7, we find that on all datasets, our top-down approach performs consistently better than both the parallel approach and the random approach. The largest improvement occurs on the Math dataset where our top-down approach saves 40% queries to achieve the best performance of the parallel approach.[9] By analyzing the *out-of-domain* ratio in Figure 5.7f, we find that our top-down approach reduces the ratio of *out-of-domain* queries by 32% on the Math dataset compared to the parallel approach. The similar pattern can also be observed on the Biology and Earth Sciences datasets where about 32% and 23% *out-of-domain* queries can be saved. For the Astronomy and Chemistry datasets, we can see that the parallel approach makes less than 20% *out-of-domain* ratios. This can explain why our top-down approach performs only slightly better than the parallel approach on the Astronomy and Chemistry datasets. However, on some of the ODP datasets, the performance curves of the parallel approach have an obvious large overlap with the random approach, while our top-down approach always outperforms the two approaches at the end of active learning.

To summarize, from our extensive experiments on eight real-world hierarchical text datasets, we empirically demonstrate that our top-down active learning framework is more effective than the state-of-the-art active learning approaches for hierarchical text classification.

[9]The top-down approach requires 600 queries to achieve 0.4 hierarchical F-measure of the parallel approach which requires 1,000 queries. The saving is (1000 – 600)/1000=40%.



(a) Astronomy (b) Biology (c) Chemistry

(d) Earth Sciences (e) Math (f) Out-of-domain Ratios

Figure 5.7: Hierarchical F-measure and the ratios of *out-of-domain* queries on the ODP datasets.

## 5.5 Summary

In this chapter, we studied the problem of using active learning to improve the performance of hierarchical text classification. A major challenge for effective hierarchical active learning is to form the unlabeled pools to avoid the so-called out-of-domain queries. Previous state-of-the-art approaches tackle this challenge by simultaneously forming the unlabeled pools on all the categories of the hierarchy regardless of the inherited hierarchical dependence of classifiers. We propose a novel top-down hierarchical active learning framework which utilizes top-down tree structure to form the unlabeled pools. Under our framework, we propose several effective strategies to tackle the out-of-domain problem and the other challenges encountered. With extensive experiments on eight real-world hierarchical text datasets, we demonstrate that our top-down framework is highly effective, and it outperforms the state-of-the-art hierarchical active learning methods by reducing 20% to 40% queries.

We believe that our top-down active learning framework can greatly help the construction of hierarchical classifiers for search engines with hierarchies, especially when the labeled examples are very costly to acquire.

# Chapter 6

# Mining Academic Topics in Universities

In previous chapters, we have discussed the foundation of building SEEU. The fundamental algorithm is the hierarchical webpage classification. Due to the resource constraints of our small cluster, we cannot conduct a full classification of the entire web and run a full search engine such as Google or Bing. However, our classification system can be very useful for organizations such as governments, universities, companies, or any vertical areas.

In this chapter, we present a novel hierarchical classification framework for mining academic topics in universities. In our framework, we train a hierarchical classifier based on the Wikipedia academic disciplines, and apply it to mine academic topics in the 12 largest universities in Ontario, Canada. According to our comprehensive experiments, the academic topics mined by our hierarchical classification framework are reasonable, consistent with the real-world topic distribution in most universities, and better than the traditional LDA topic modeling approaches. To the best of knowledge, this is the first work that mines academic topics in large-scale university webpage dataset with satisfying results.

## 6.1 Introduction

Academic research is one of the major activities in universities. It plays an important role in facilitating knowledge discovery and technology innovation in modern society. Mining academic topics from university webpages can help researchers analyze the organization and relations of academic research in universities. It is also the basic of university based web applications, such as the university web search engines with hierarchies (i.e., SEEU).

A simple approach to mine academic topics in universities is to use the department or faculty organization structure (i.e., a web directory in a university) [80], and classify webpages by matching department host names in URLs. However, such a simple approach can only extract a very shallow hierarchy where specific and more useful topics are missing. For example, at Western, although we can judge webpages belonging to the topic of “Computer Science” (CS) by matching the CS department host name (`csd.uwo.ca`) in URLs, we may not know if a webpage belongs to specific topics, such as “Support vector machine” or “Computational linguistics”, because topic related information is often not explicitly expressed in URLs. In addition, this approach can only assign hard classification to webpages, and not degrees of confidence (or probability), which is very important for ranking webpages in an academic topic hierarchy (as discussed in Chapter 4).

Some content based methods have been proposed for mining topics in text documents. The most popular methods are the topic modeling approach in natural language processing and information retrieval research, such as LSA [34], PLSA [54] and LDA [11]. Basically, these methods use mathematical or statistical algorithms to find topics by analyzing the co-occurrence of word-document data (see detailed review in Chapter 2.4). A major advantage of these approaches is that they do not require human supervision. Thus, the topic mining process can be conducted automatically and directly on the targeted dataset. This is based on the assumption that the topic structure can be self-described (or generated) directly by the dataset. However, the topics generated by these methods are heavily dependent on the quality of text corpora. For webpage collections that contain a lot of heterogeneous data (e.g., documents with different length and format, skewed topic distribution and so on), the topic quality of these methods may not be satisfying. This is confirmed in our experiments on the SEEU dataset where the mined academic topics by these approaches are not satisfying (see Section 6.6).

In this chapter, we present a novel hierarchical classification framework for mining academic topics in universities. In our framework, we build an academic topic hierarchy based on the commonly-accepted Wikipedia academic disciplines[1]. Based on this academic topic hierarchy, we train a hierarchical classifier, and apply it to classify webpages from the top 12 largest universities in Ontario, Canada into the topic hierarchy. According to our comprehensive experiments, the academic topic pattern mined by our hierarchical classification system is reasonable, consistent with the common sense of topic distribution in these universities, and better than the state-of-the-art LDA topic modeling approach. To the best of knowledge, this is the first work that uses hierarchical classification to mine academic topics in large-scale university webpage datasets with satisfying results.

The rest of this chapter is organized as follows. In Section 6.2, we present a high-level overview of our general framework for mining topics by hierarchical classification. The detailed components of the framework will be discussed in the following three sections. Specifically, Section 6.3 discusses the methods to build topic hierarchies. Section 6.4 shows the training of the hierarchical classifier. Section 6.5 describes the SEEU webpage dataset and conducts a comprehensive analysis of the academic topics mined by our hierarchical classification system. In Section 6.6, we compare our approach to a state-of-the-art topic modeling approach. The last section is the summary.

## 6.2 The General Framework

In this section, we describe our general framework for mining topics by hierarchical classification in domain-specific datasets, such as the university webpage dataset. Figure 6.1 shows the high-level overview of our framework which consists of three steps:

1. Building a topic hierarchy based on the application domain knowledge.

2. Training a hierarchical classifier for the hierarchy.

[1]http://en.wikipedia.org/wiki/List_of_academic_disciplines

3. Conducting hierarchical topic classification and visualizing/analyzing the results.

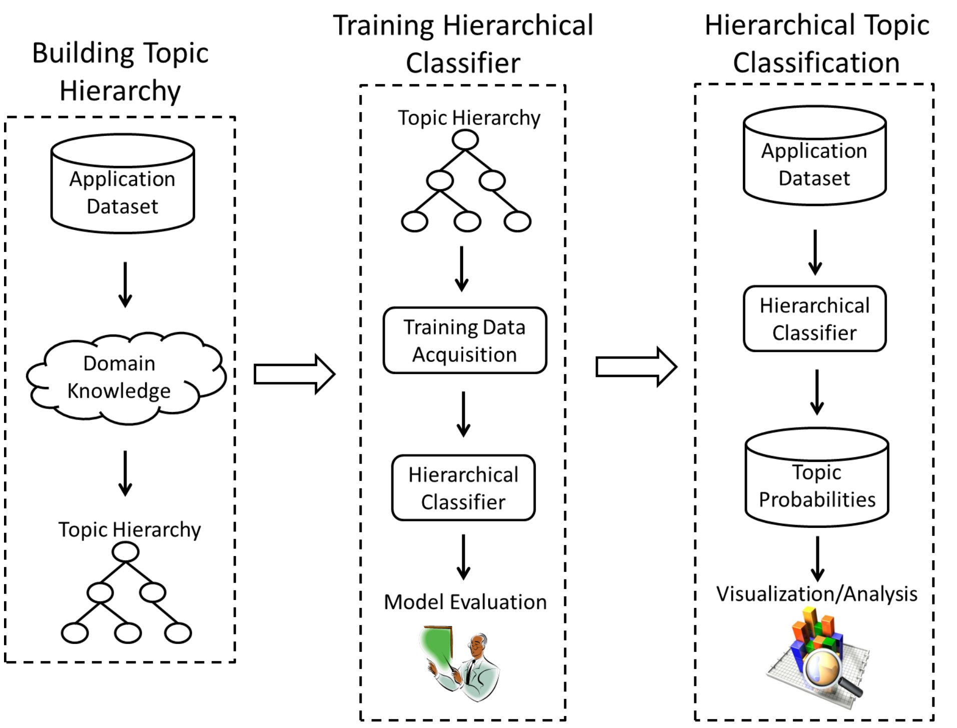


Figure 6.1: The general framework for mining topics by hierarchical classification.

The first and most important step in our framework is to build a reasonable and commonly-accepted topic hierarchy for the application domain. In this step, we emphasize the importance of the domain knowledge. Usually, for well-developed application domains (e.g., electronic encyclopedia, web directory, patent, news media), there already exists some high-quality topic taxonomies on the web, such as the Wikipedia topic hierarchy[2], the ODP (Open Directory Project) category hierarchy[3], the WIPO (World Intellectual Property Organization) patent taxonomy[4], and the IPTC (International Press and Telecommunications Council) NewsCodes taxonomy[5]. Due to the large number of human editing effort, the quality of these topic hierarchies would be much better than the topic hierarchies extracted by ad-hoc methods or tools. Thus, we can directly build the topic hierarchy by reusing these existing taxonomies. Section 6.3 discusses the details of our topic hierarchy for the domain of universities.

The second step is to train a hierarchical classifier for the pre-built topic hierarchy. In our framework, we use supervised machine learning algorithms, more specifically, the hierarchical classification algorithm (see Chapter 3), to train the classifier. The most difficult part in this step is to collect enough training data for the learning algorithm. The training data should be consistent with the targeted application dataset and large enough to cover sufficient words and

[2]http://en.wikipedia.org/wiki/Category:Main_topic_classifications
[3]http://www.dmoz.org/
[4]http://web2.wipo.int/ipcpub
[5]http://iptc.cms.apa.at/site/NewsCodes/

documents related to the application domain. After the training data acquisition, we can train a hierarchical classifier. Finally, it is very important to perform model evaluation to assess the quality of the trained hierarchical classifier. We discuss our data acquisition method and the evaluation of the trained hierarchical classifier in Section 6.4.

The third step is the standard text classification task where we apply the trained hierarchical classifier to classify the application dataset into the pre-built topic hierarchy. After that, we obtain topic probabilities of each text document and conduct visualization analysis to reveal the mined topic pattern. Section 6.5 applies the trained hierarchical classifier on a large-scale university webpage dataset and give a detailed discussion about the topic pattern mined by our framework.

## 6.3 Building Academic Topic Hierarchy

A reasonable and commonly accepted topic hierarchy is very important for the university search engine. It can greatly benefit the search experience of users. There are several criteria that we think a good topic hierarchy should have for university search engines.

1. **Academic Topics**. Academic institutions usually have various academic disciplines. It is important to define a hierarchy with a broad coverage of common academic disciplines so that the majority of university users will feel familiarized and convenient to find the desired topics.

2. **Tree structure**. The tree-structured hierarchies are intuitive and user-friendly user interfaces for most of web applications. However, a complicated hierarchical structure (with deep categories) may not be easy for browsing. Thus, the topic hierarchy in SEEU should not be too deep.

In fact, to build the hierarchy for SEEU, we can reuse existing well-known taxonomies, such as the Open Directory Project (ODP)[6] and the category structure in *Wikipedia*.

- ODP hierarchy. The topic hierarchy of ODP is a tree structure. The relationship between a parent category and a child category basically follows the ”IS-A” constraint. Thus, it may be a good choice to use the ODP taxonomy for SEEU. Several works [67, 92] have successfully used the ODP hierarchy in text classification. However, we argue that the ODP hierarchy is not suitable for academic institutions, because ODP is a general (not academe oriented) taxonomy. For example, we can find many non-academic categories in the ODP, such as “Shopping”, “Home”, “Games”, “Clothing”, “Vehicles” and “Massage Equipment”. Those (sub)categories are useless for university search engines. Moreover, we can also observe a noisy mixture of academic and non-academic subcategories inside ODP. For example, under the top-level category “Computers”, we can find non-academic subcategories of “E-Books”, “Usenet” and “Emulators” as well as academic subcategories, such as “Parallel Computing” and “Artificial Intelligence”. Such a confusing category structure will not only mislead users, but also reduce the classification performance.

[6]http://www.dmoz.org

- Wikipedia hierarchy. The taxonomy in Wikipedia is a directed acyclic graph (*DAG*) rather than a tree-structured hierarchy. In a DAG graph, one node can have multiple parent nodes. For example, in Wikipedia, the category "Universe" has two parent categories, i.e., "Nature" and "Astrophysics". Such a DAG hierarchy violates the second criterion (tree structure) we mentioned before. Therefore, it may confuse normal users who are usually familiar with the tree-based browsing. Another problem of Wikipedia categories is that not all the child-parent relations strictly follow the "IS-A" semantics. For example, on the help page of Wikipedia categories[7], there is a category path "History" → "History by location" → "History by country" → "History of Australia" → "History of Australia by location" → "History of Australia by state or territory" → "New South Wales, Queensland". We can see that, the state "New South Wales" definitely does not belong to (is a) "History".

Therefore, in this thesis, we prefer to manually[8] build an academic topic hierarchy. We build an academic topic hierarchy by using the Wikipedia "List of Academic Disciplines"[9], which is a tree-structured hierarchy containing a broad scope of academic disciplines. We also survey and adopt categories of web directories in several top Medical Doctoral universities in Canada (e.g., Western, Queen's and Toronto). After several rounds of discussion with domain experts (i.e., faculty and graduate students) in different disciplines, we eventually build a four-level academic topic hierarchy of 464 categories. It covers a broad scope of academic disciplines, including "Business", "Education", "Engineering", "Health science", "Humanities", "Journalism and media studies", "Law", "Medicine", "Natural sciences" and "Social sciences".

A partial snapshot of this academic topic hierarchy can be seen in Figure 6.2. To help readers understand how this hierarchy works in SEEU, we also show an example that searches for "data mining" under the category hierarchy "Natural sciences"→"Computer sciences"→"Information science" in SEEU (see Figure 6.3). It should be mentioned that although there may exist some categories not covered by our hierarchy, it does not mean that the university webpages belonging to those categories are lost. In SEEU, users can still find those pages by simply searching for keywords without any categories or searching for keywords at a higher-level category (e.g., "Computer sciences") .

## 6.4 Training the Hierarchical Classifier

In this section, we describe the training of the hierarchical classifier in our framework. We have already discussed the hierarchical classification algorithm in Chapter 3. We will mainly focus on the training data acquisition and model evaluation of the trained hierarchical classifier.

### 6.4.1 Training Data acquisition

After we build the topic hierarchy, we need to collect enough labeled documents as training set. The dataset we collect must satisfy two criteria. Firstly, the text representation of the training

[7]`http://en.wikipedia.org/wiki/Help:Categories`
[8]We still use existing topic hierarchies, such as Wikipedia as basic.
[9]`http://en.wikipedia.org/wiki/List_of_academic_disciplines`

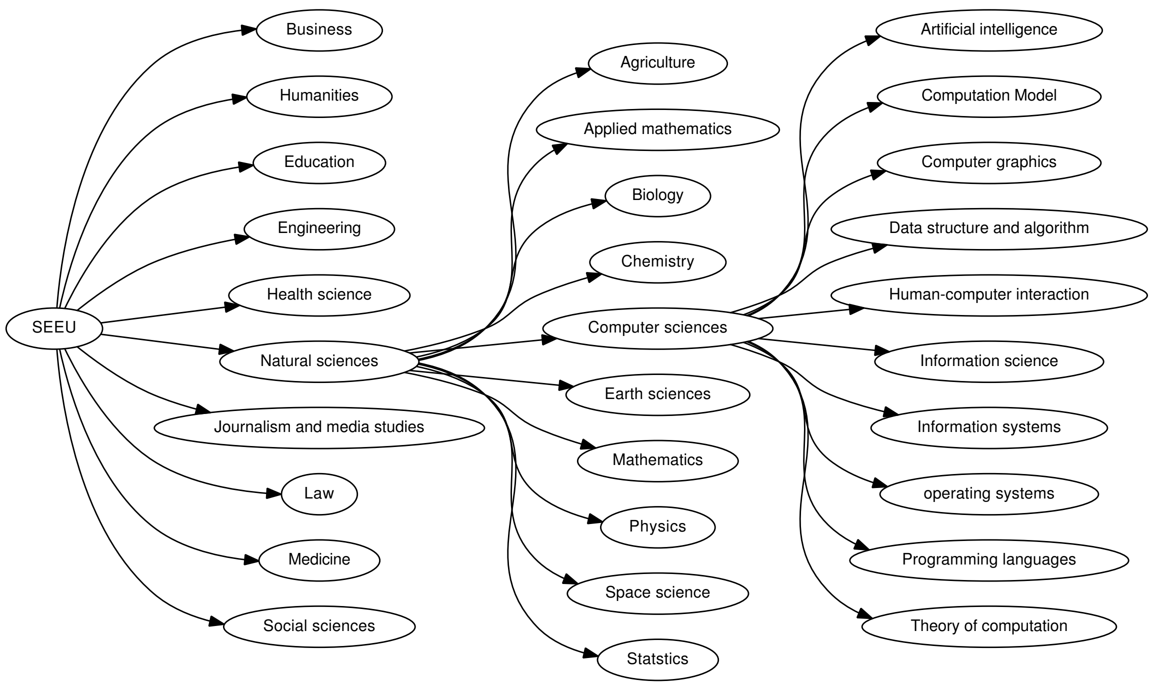


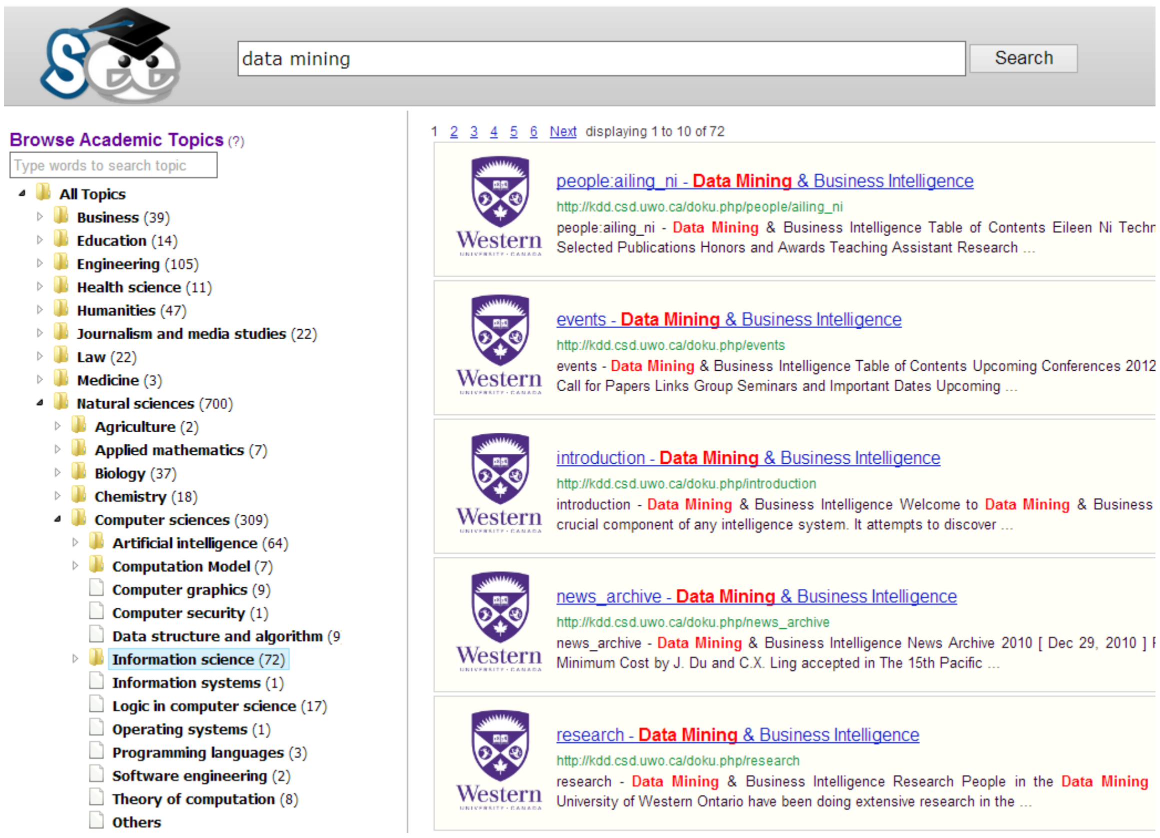
Figure 6.2: A partial hierarchy in SEEU under the "Natural sciences" category.

documents should be as similar as the targeted university webpages. Secondly, the collected dataset should contain accurate yet enough documents.

One may say that as we build the topic hierarchy based on the Wikipedia academic disciplines, we can simply collect the Wikipedia pages in those categories as the training set. We argue that this is actually not a good idea. For most Wikipedia categories, the number of pages belonging to them is so small that it is insufficient to train accurate text classifiers. Moreover, the text representation of Wikipedia articles is quite different from normal university webpages. The difference of text distribution may even degrade the classification performance of supervised learning algorithms.

To tackle this problem, we propose to use commercial search engines to collect the labeled webpages for each category of SEEU's topic hierarchy. This method is also used in Li [77] and Ha-Thuc [47]. Specifically, for each category, we firstly carefully prepare a list of keywords that are closely related to the meaning of the categories (see Table 6.1 for an example). We then submit these keywords to several commercial search engines (e.g., Google or Bing). The top ranked webpages can be approximately treated as labeled examples for a category.[10] By automatically exploring the huge number of indexed webpages in commercial search engines, we can easily collect a large number of labeled webpages.

[10]To reduce the number of noisy webpages, we only use the top 100 returned results which are usually relevant to the query.



Browse Academic Topics (?)

Type words to search topic

- All Topics
  - Business (39)
  - Education (14)
  - Engineering (105)
  - Health science (11)
  - Humanities (47)
  - Journalism and media studies (22)
  - Law (22)
  - Medicine (3)
  - Natural sciences (700)
    - Agriculture (2)
    - Applied mathematics (7)
    - Biology (37)
    - Chemistry (18)
    - Computer sciences (309)
      - Artificial intelligence (64)
      - Computation Model (7)
      - Computer graphics (9)
      - Computer security (1)
      - Data structure and algorithm (9
      - Information science (72)
      - Information systems (1)
      - Logic in computer science (17)
      - Operating systems (1)
      - Programming languages (3)
      - Software engineering (2)
      - Theory of computation (8)
      - Others

1 2 3 4 5 6 Next displaying 1 to 10 of 72

people:ailing_ni - **Data Mining** & Business Intelligence
http://kdd.csd.uwo.ca/doku.php/people/ailing_ni
people:ailing_ni - **Data Mining** & Business Intelligence Table of Contents Eileen Ni Techr
Selected Publications Honors and Awards Teaching Assistant Research ...

events - **Data Mining** & Business Intelligence
http://kdd.csd.uwo.ca/doku.php/events
events - **Data Mining** & Business Intelligence Table of Contents Upcoming Conferences 2012
Call for Papers Links Group Seminars and Important Dates Upcoming ...

introduction - **Data Mining** & Business Intelligence
http://kdd.csd.uwo.ca/doku.php/introduction
introduction - **Data Mining** & Business Intelligence Welcome to **Data Mining** & Business
crucial component of any intelligence system. It attempts to discover ...

news_archive - **Data Mining** & Business Intelligence
http://kdd.csd.uwo.ca/doku.php/news_archive
news_archive - **Data Mining** & Business Intelligence News Archive 2010 [ Dec 29, 2010 ] F
Minimum Cost by J. Du and C.X. Ling accepted in The 15th Pacific ...

research - **Data Mining** & Business Intelligence
http://kdd.csd.uwo.ca/doku.php/research
research - **Data Mining** & Business Intelligence Research People in the **Data Mining**
University of Western Ontario have been doing extensive research in the ...

Figure 6.3: Searching "data mining" under the topic hierarchy "Natural sciences"→"Computer sciences"→"Information science" in SEEU.

Table 6.1: A sample of the manually prepared phrases for querying search engines.

| NO. | Category | Query keywords |
|---|---|---|
| 1 | Business | "Business","Business analysis","Marketing" |
| 2 | Education | "Education", "Curriculum instruction", "Education technology" |
| 3 | Engineering | "Engineering", "Engineering research", "Engineering system" |
| 4 | Health science | "Health science" |
| 5 | Humanities | "Humanities", "Arts", "Human culture" |
| 6 | Journalism and media studies | "Journalism", "Media studies", "Communication" |
| 7 | Law | "Law" |
| 8 | Medicine | "Medicine", "Disease", "Diagnosis and treatment" |
| 9 | Natural sciences | "Natural sciences", "Formal sciences" |
| 10 | Social sciences | "Social sciences", "Society", "Sociology" |

In addition, as we are building a search engine for academic institutions (not the general web), we restrict the search results in the domain of universities by adding "site:edu" in the query.[11] Thus, the difference of data distribution between the labeled dataset (i.e., the collected webpages from search engines) and the testing set (i.e., the university webpages) can be greatly reduced. Readers may ask why not just search webpages in the targeted universities? We argue that this may cause overfitting problem [36] as we will train and test the classification model on the same dataset.

It should be noted that we can leverage the hierarchical "IS-A" relations to further enlarge the training set for intermediate categories by propagating the returned webpages to lower-level subcategories. For example, when we send a query "Artificial intelligence" to search engines, we can lift the returned webpages to its ancestor categories, i.e., "Computer sciences" and "Natural sciences".

By crawling search engines, we eventually collect about 570,000 labeled webpages in 464 categories as the training set. We randomly sample 30 categories to assess the quality of the dataset. Two graduate students were recruited to manually examine a small subset (less than 500) of the webpages in each category, and reported the average accuracy of labeling webpages. Based on their report, our dataset has about 90% accuracy which we believe is acceptable for training SVMs due to the soft margin principle of SVM which is very robust to noisy data[26].

We use the bag-of-words model to represent the crawled webpages. Each webpage is treated as a vector of word features represented as TF·IDF weights. After we extract text features by our feature extraction tool (see Chapter 3), we remove rare words occurring in less than three documents and apply Porter Stemming [89] to all words. Short documents with less than ten words are also removed. Eventually, we generate 563,163 labeled examples in 464 categories as the training set. We tabulate the statistics of our training set in Table 6.2.

Table 6.2: The statistical information of the training set at each level of SEEU's hierarchy.

| Level | Categories | Class Ratio | Examples | Features |
|---|---|---|---|---|
| 1 | 10 | 0.0979 | 563,163 | 1,329,110 |
| 2 | 88 | 0.1009 | 57,777 | 432,697 |
| 3 | 261 | 0.1012 | 15,362 | 254,037 |
| 4 | 105 | 0.1717 | 5,788 | 163,521 |

Based on the collected labeled examples for the new hierarchy, we can train a hierarchical classification system. We use search engines to collect the training set. It may be argued that this method will introduce noise to the dataset, and result in a bad hierarchical classification system. However, in the next subsection of model evaluation, we will show that the trained hierarchical classification system is actually very good.

### 6.4.2 Evaluation of Classification Model

In this subsection, we will firstly conduct experiments to evaluate the classification performance of our hierarchical classification system on the collected dataset. After that, we will analyze the classification models to explain its effectiveness.

[11] Both Google and Bing support "site" syntax in the queries

**Cross-validation Experimental Results**

We use the hierarchical SVM classifiers developed in Chapter 3 as the classification model. As the feature size is very large (i.e., more than one million words in Table 6.2), we need to select a relatively small number of features to reduce training time. We use the DF (Document Frequency) feature selection algorithm and 50,000 features in this experiment. They are the best parameter setting in our previous hierarchical classification experiments (see Chapter 3.3).

Five-fold cross validation is performed on the dataset for evaluation. We show plots of the average precision, recall and F1-score at each level of the hierarchy in Figure 6.4. We can see that the hierarchical SVM classifiers can achieve a decent performance on the collected dataset. Specifically, the average F1-score at the top two levels is larger than 0.7. This is due to the good balance between high precision and high recall (i.e., both are larger than 0.7). For the 3rd and the 4th levels, although the F1-score drops, it should be noted that the worst F1-score is still larger than 0.6 which is better than our previous experiments on the ODP dataset.

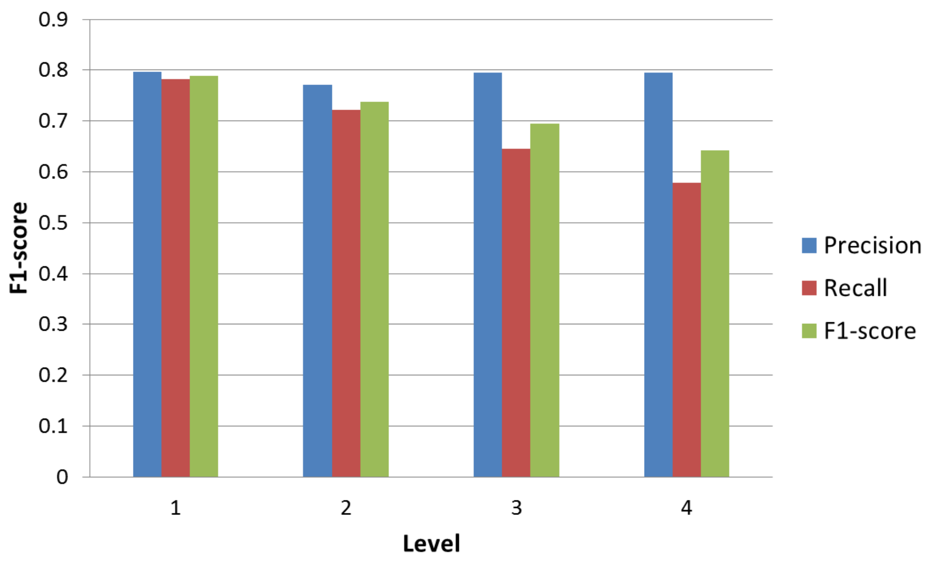


Figure 6.4: The average F1-score at different levels by the hierarchical SVM classifiers on the SEEU dataset.

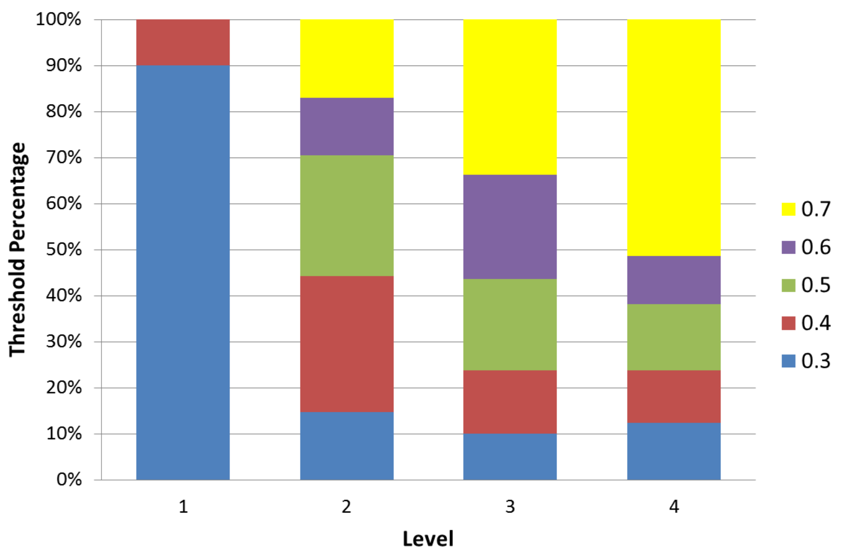


Figure 6.5: The optimal threshold distribution at different levels.

We study the learned prediction thresholds. Figure 6.5 shows the threshold distribution at each level. We can see that the result is consistent with our previous hierarchical classification experiments in Chapter 3.3. Generally speaking, most categories at the top two levels favor lower thresholds (i.e., less than or equal to 0.5) while larger thresholds (i.e., 0.6 and 0.7) are more often used at the lower levels. This explains the high recall at the top levels and the high precision at the deeper levels.

## Interpreting Hierarchical Classification Models

Why can our hierarchical SVM classifiers can achieve such a good performance despite the potentially noisy training set? To explain the effectiveness of our hierarchical SVM classifiers, we use a white-box method to analyze the learned SVM models. Recall that a linear SVM classification model is in the form of linear functions (see Chapter 2.1.3),

$$f(x; w) = \vec{\mathbf{w}} \cdot \vec{\mathbf{x}} = \sum_{1 \le i \le n} w_i x_i \tag{6.1}$$

where $x_i$ is a word in the document $x$; $w_i$ is the learned weight for that word. Intuitively, for each word $x_i$, a large positive weight of $w_i$ means that the word $x_i$ may be quite relevant to the learned topic while a negative weight may indicate an irrelevant word. Thus, we can actually analyze the words with highest weights to interpret the classification models of linear SVMs.

We firstly analyze the words with highest weights in SVM models at the first-level topics. We tabulate the top ten words in Figure 6.6. We can see that those words are quite relevant to the learned concept (topics). For example, in the topic of "Business", the top ten words are all about the business activities or related areas, such as *entrepreneurship*, *purchasing* and *trade*. As SVM models successfully learn those domain-specific words, it makes the hierarchical classifier achieve a high *recall* as shown in Figure 6.4. In addition, we also find that there is no overlapping of the top ten words between each pair of topics. Even by checking the top

| **Business** | **Education** | **Engineering** | **Health science** | **Humanities** |
|---|---|---|---|---|
| business | education | engineering | kinesiology | philosophy |
| insurance | vocational | assurance | clinical | morphology |
| entrepreneurship | counselor | bioengineering | pharmaceutical | history |
| risk | bilingual | control | optometry | bioethics |
| finance | alternative | vlsi | rehabilitation | syntax |
| purchasing | cooperative | telecommunications | histology | ergonomics |
| trade | peace | automotive | hepatology | literature |
| bargaining | higher | polymer | dentistry | rhetoric |
| marketing | distance | petroleum | pharmacy | speaking |
| resources | critical | aerospace | nutrition | storytelling |

(a) First five topics

| **Journalism** | **Law** | **Medicine** | **Natural sciences** | **Social sciences** |
|---|---|---|---|---|
| newspaper | paralegal | physiology | endocrinology | social |
| magazine | contract | epidemiology | decision | political |
| translation | law | dietetics | information | studies |
| propaganda | jurisprudence | primary | multimedia | psychology |
| advertising | criminal | infectious | anatomy | sociology |
| intercultural | tax | oncology | knowledge | anthropology |
| television | forensic | general | statistics | ethnography |
| community | tort | cardiology | algebra | slavic |
| radio | corporations | psychiatry | biology | geography |
| technical | competition | surgery | virology | econometrics |

(b) Last five topics

Figure 6.6: The top ten most relevant words in SVM models at the first-level topics.

100 words, we find that the average overlapping rate is still less than 1% (i.e., one word). This means that our classification system can learn very good discriminative words among different concepts. This explains why the *precision* is also very high in Figure 6.4.

One may argue that it may be easy to learn the first-level topics as they are general concepts. Will our classification system also learn good models for deeper and more specific topics? As we are more familiar with "Computer science" research, we choose the subtopics under "Computer science" as a case study. There are 13 subtopics under it. Figure 6.7 tabulates the top ten most relevant words for those subtopics. We can see that, overall, the results are also very good. For example, we can find familiar words such as *data mining*, *information retrieval* and *knowledge discovery* in "Information science"; *object oriented* in "Programming languages" and *deadlock* and *os* in "Operating systems".

We check the F1-score for those subtopics under "Computer science". We find that most of subtopics (12/13) have F1-score around 0.69, similar to the average F1-score (0.69) at the third level. It means that the SVM models learned under "Computer science" are also reasonbly

| **Artificial Intelligence** | **Computer graphics** | **Computation Model** | **Computer security** | **Data structure and algorithm** |
|---|---|---|---|---|
| artificial | graphic | cloud | cryptography | algorithms |
| machine | image | wireless | security | structure |
| vision | visual | high | tolerance | parallel |
| robotics | scientific | quantum | fault | data |
| experts | opengl | grid | cipher | structural |
| natural | processing | ubiquitous | incident | distribution |
| decision | process | performance | cryptographic | hash |
| cognitive | siggraph | parallel | encryption | random |
| learning | signal | distribution | firewall | array |
| logistics | pixels | hpc | virus | adt |

(a) First five subtopics

| **Human -computer interaction** | **Information science** | **Information systems** | **Logic in computer science** | **Operating systems** |
|---|---|---|---|---|
| hci | management | technology | logic | operations |
| human | knowledge | information | fuzzy | system |
| interaction | multimedia | system | semantics | systems |
| hcii | database | systems | plc | deadlock |
| interface | information | telecommunications | circuit | os |
| interactive | retrieval | business | lambda | cpu |
| usable | mining | services | alps | memory |
| chci | data | desk | syntax | kernel |
| gesture | science | gis | programme | silberschatz |
| multimodal | discovery | mi | iff | operating system |

(b) Second five subtopics

| **Programming languages** | **Software engineering** | **Theory of computation** |
|---|---|---|
| program | software | automata |
| language | engineering | theory |
| concurrent | sei | computer |
| orientation | se | symbols |
| compiled | softwareengineering | algebra |
| oriented | cmmi | turing |
| functional | swe | complex |
| function | swen | symbolic |
| linguistics | mse | sipser |
| object | sepg | np |

(c) Last three subtopics

Figure 6.7: The top ten most relevant words in SVM models at the subtopics under “Computer sciences”.

good. However, there is one subtopic with only 0.55 F1-score. It is the “Information systems” category. We can see from Figure 6.7(b) that several words (e.g., *desk*, *services* and *system*) are too general to represent the concept of “Information systems”. On the other hand, it may also due to the reason that the concept of “Information systems” is too broad compared with other subcategories and thus difficult to learn.

To summarize, in this section, we have demonstrated that our hierarchical SVM classifiers can achieve good performance for academic topic classification in the collected dataset. In the next section, we will apply it to mine academic topics from the challenging two million university webpages in the SEEU dataset.

## 6.5 Hierarchical Topic Classification

In this section, we describe the SEEU university webpage dataset and apply the trained hierarchical classification system to mine (classify) academic topics in the SEEU dataset.

### 6.5.1 SEEU Webpage Dataset

According to Wikipedia, there are in total 98 universities in Canada.[12] Due to the limited computational resources, we can not index and classify webpages from all the universities in Canada. In this thesis, we focus on the 12 largest universities in Ontario based on the total number of student enrolment. They are Toronto, York, Ottawa, Western, Ryerson, McMaster, Carleton, Waterloo, Guleph, Queen’s, Brock and Windsor.

For each university, we use the popular *Apache Nutch*[13] crawler to crawl the webpages in its university domain (see detailed crawling method in Appendix A), and use our feature extraction tool to extract the text in webpages. We use the same feature preprocess method (see Section 6.4) to clean the dataset. The final SEEU university dataset is tabulated in Table 6.3.

[12]See `http://en.wikipedia.org/wiki/List_of_universities_in_Canada`.
[13]The home page of *Nutch* is `http://nutch.apache.org/`.

Table 6.3: The 12 largest universities in Ontario based on the total number of student enrolment. Enrolment data are from the Wikipedia page “List of universities in Canada” in April 2013.

| University | URL | Enrolment | Pages | Unique Words | Total words |
|---|---|---|---|---|---|
| Toronto | `www.utoronto.ca` | 74,760 | 433,418 | 606,865 | $3.23 \times 10^8$ |
| York | `www.yorku.ca` | 52,290 | 214,142 | 471,077 | $1.46 \times 10^8$ |
| Ottawa | `www.uottawa.ca` | 38,700 | 122,708 | 284,777 | $0.87 \times 10^8$ |
| Western | `www.uwo.ca` | 34,100 | 278,414 | 394,721 | $1.35 \times 10^8$ |
| Ryerson | `www.ryerson.ca` | 31,770 | 65,900 | 156,710 | $0.28 \times 10^8$ |
| McMaster | `www.mcmaster.ca` | 26,070 | 261,256 | 404,449 | $1.48 \times 10^8$ |
| Carleton | `www.carleton.ca` | 24,250 | 63,012 | 214,691 | $0.38 \times 10^8$ |
| Waterloo | `www.uwaterloo.ca` | 24,160 | 221,283 | 466,409 | $2.04 \times 10^8$ |
| Guleph | `www.uoguelph.ca` | 22,080 | 85,239 | 192,245 | $0.45 \times 10^8$ |
| Queen’s | `www.queensu.ca` | 20,550 | 153,291 | 370,970 | $1.24 \times 10^8$ |
| Brock | `www.brocku.ca` | 17,006 | 43,064 | 154,155 | $0.88 \times 10^8$ |
| Windsor | `www.uwindsor.ca` | 16,180 | 34,417 | 140,141 | $0.17 \times 10^8$ |

### 6.5.2 Results from Hierarchical Classification

We apply the trained hierarchical classification system to classify webpages in the SEEU dataset into the academic topic hierarchy, and analyze the results.

We briefly describe the experimental methodology in this data mining task (step 3 in our general framework; see Section 6.2). Specifically, for each webpage in the dataset, we apply our trained hierarchical SVM classifiers to output a vector of probabilities indicating the degree of confidence of each topic in the hierarchy.[14] Based on these topic probability vectors, we are interested in answering the following questions.

1. How are the academic topics distributed in universities?
2. What relations exist between different topics?
3. Which university has the largest program in a specific topic?

In this analysis, as we are more familiar with the academic disciplines of “Natural sciences” and especially the “Computer sciences” research, we will mainly analyze the topic classification results under the topic hierarchy of “All” → “Natural sciences” → “Computer sciences”.

[14]To preserve the hierarchical information, we adjust the topic probability prediction outputted by SVMs as $P_{adj}(t) = P(t) \times \prod_{t' \in \Uparrow t} P(t')$, where $P(t)$ is the likelihood probability of topic $t$ and $\Uparrow t$ is the set of all the ancestor topics of the topic $t$.

**Topic Pattern at the First Level**

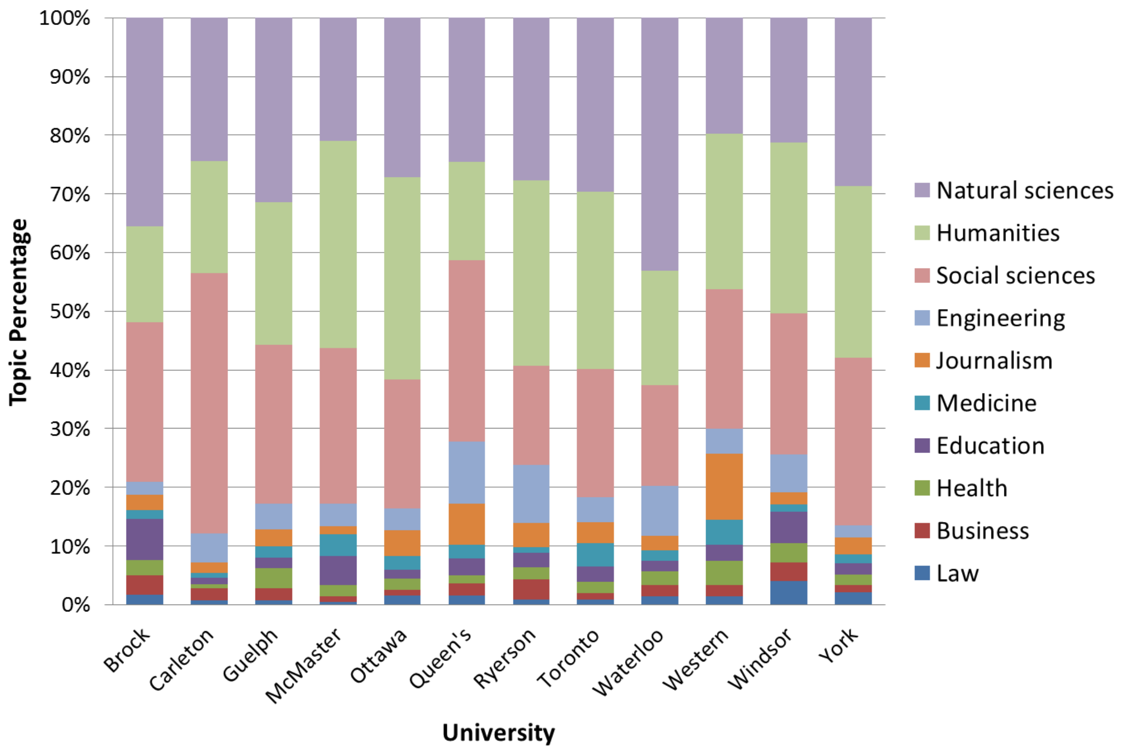


Figure 6.8: Topic classification distribution at the first level.

We first analyze the topic classification results at the first level of the topic hierarchy. For each university, we plot the average topic probabilities of all the webpages in Figure 6.8. Each stacked column represents the first-level topic classification distribution (percentage) in a university. We can see a clear pattern that all the universities have a large portion (about 80%) of webpages categorized into three primary topics, i.e., “Natural sciences”, “Humanities” and “Social sciences”. This result is reasonable because most of the scientific research works in universities are related to the three topics. Successfully identifying the three important topics is a strong evidence of the effectiveness of our classification framework. The remaining 20 percentage of webpages are categorized into seven small topics. By the percentage ranking, they are “Engineering”, “Journalism”, “Medicine”, “Education”, “Health”, “Business” and “Law”. This ordered topic list is also reasonable. We can see that the top ranked topics are also about research, such as “Engineering” while the low ranked topics (e.g., “Business” and “Law”) are mainly about professional education where relatively fewer webpages are published in universities.

**Topic Pattern under “Natural sciences”**

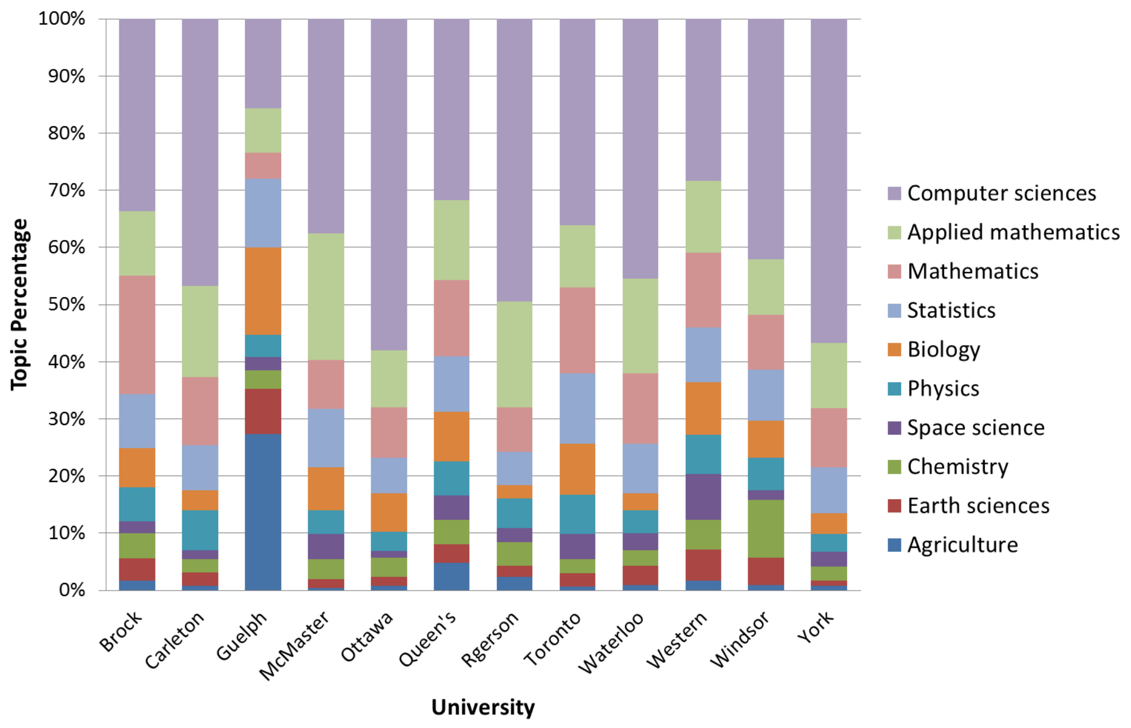


Figure 6.9: Topic classification distribution under “Natural sciences”.

Next, we study the topic classification results under “Natural sciences”. From Figure 6.9, we can see that the top four largest topics are disciplines concerning formal science[15], such as “Computer sciences”, “Mathematics” and “Statistics”, while the rest topics are about classical natural science research, such as “Physics” and “Chemistry”. Is this classification pattern by our hierarchical classification system consistent with the real world webpage distribution? To study this issue, we use commercial search engines to estimate the number of webpages in the corresponding departments in each university. For example, for the Department of Computer Science at Western, we can type “site:csd.uwo.ca” in *Google* to estimate its total number of webpages.[16] By using this method, we make an approximation of the corresponding topic order in Western as *Computer Science* > *Mathematics* > *Statistics* > *Applied Mathematics* > *Physics* > *Biology* > *Earth Science* > *Chemistry*.[17] Despite a slight difference of several pairs of topics, we can see that the general partition of formal science and natural science is consistent with ours.

It is also interesting to study the relations of these topics under “Natural sciences”. In this analysis, we still focus on the topic prediction results in Western. For the ten topics under “Natural sciences”, we can find eight corresponding departments in the Faculty of Science of

[15]`http://en.wikipedia.org/wiki/Formal_science`

[16]It should be noted that some non-academic pages may be over-counted.

[17]The “Space science” webpages are included in the *Physics* department. Western does not have “Agriculture” department.

Western. We compute the average topic probabilities for webpages belonging to these departments, and plot a heat map matrix[18] in Figure 6.10.

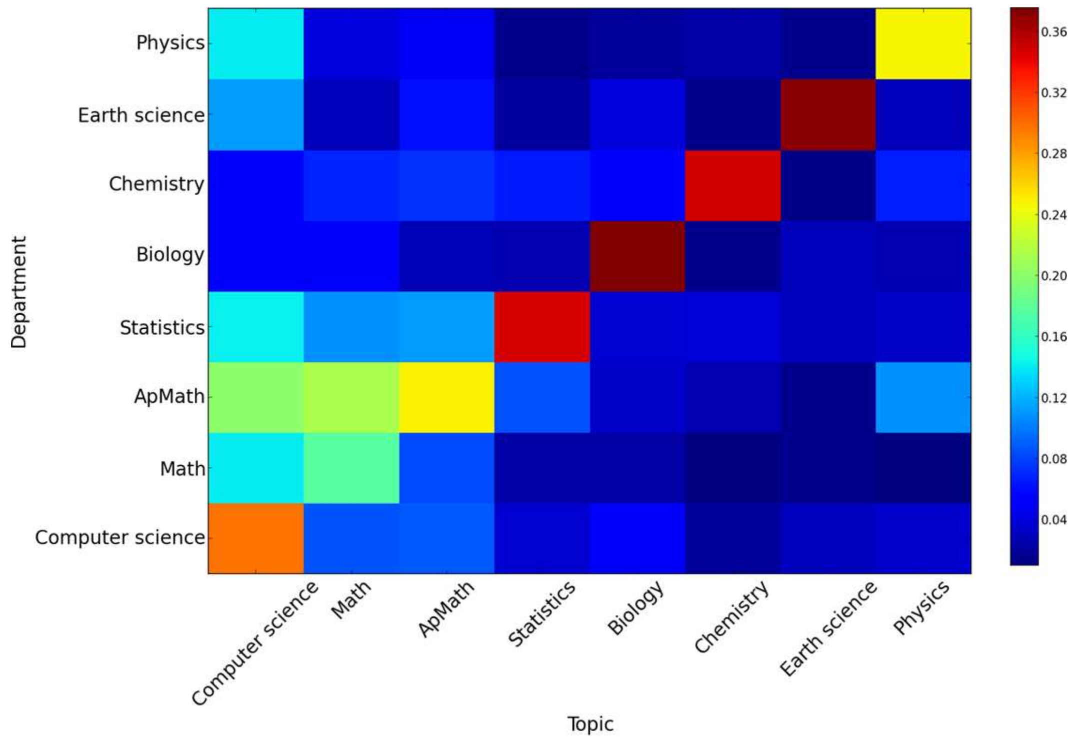


Figure 6.10: A heat map matrix showing the average topic probabilities estimated by our hierarchical classifiers for webpages belonging to departments in the Faculty of Science of Western. Each row is a vector of topic probabilities. The dark blue color means small probability while the bright red color means large probability.

We can find three interesting relations from Figure 6.10 that support the effectiveness and usefulness of our hierarchical classification framework.

- A strong diagonal in the heat map. This indicates that most webpages in those departments (at the y-axis) have high probabilities towards the corresponding inferred topics (at the x-axis), but low probabilities in other topics. It means that our hierarchical classifier can achieve good classification performance under "Natural sciences".

- A bright submatrix at the lower-left corner. This submatrix shows the relations among four closely related departments (Computer sciences, Math, ApMath and Statistics) that all deal with computation or data analysis and processing related research.

- Popularity of computer science topics. We can see another pattern that the computer science topic exists in most of the departments. It is well-known that the computer science has become an important tool to support scientific research in most research disciplines.

[18]A heat map matrix is a graphical representation of matrix where each grid shows the intensive of values.

**Topic Pattern under "Computer sciences"**

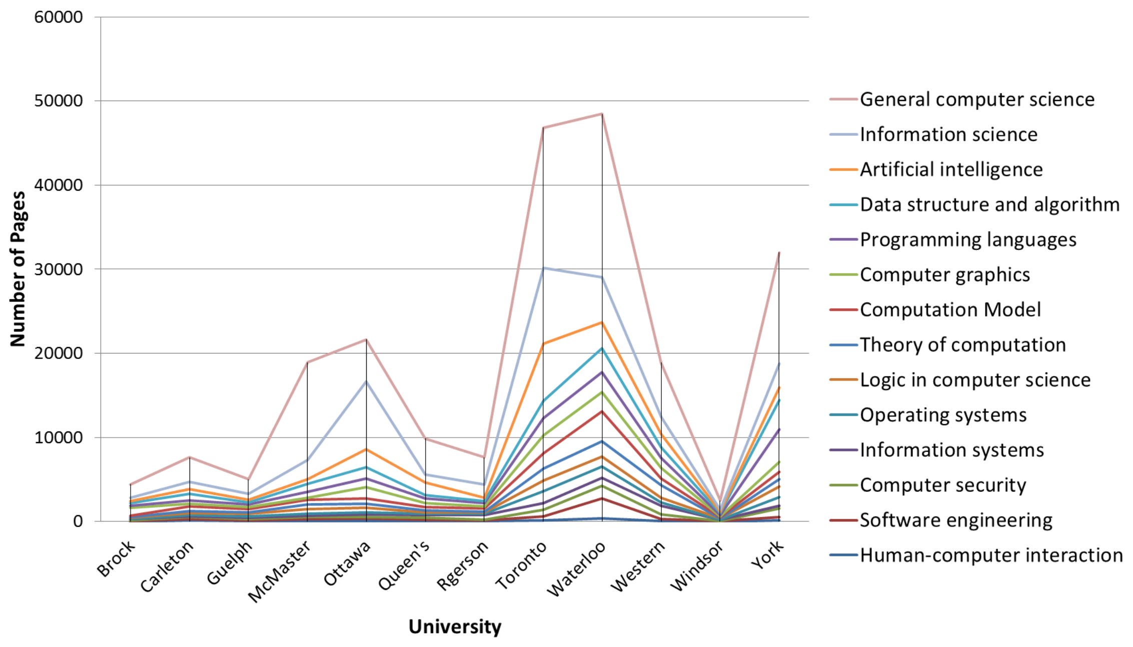


Figure 6.11: Number of pages for topics under "Computer sciences".

Finally, we study the detailed topic classification results under "Computer sciences" (CS). Different to the previous analysis that focuses on the topic classification distribution, in this study, we analyze the amount of research activity by visualizing the number of predicted CS webpages of each university in Figure 6.11.

We analyze the ranking of page number in Figure 6.11. This ranking is consistent to the real-world ranking by scale and size of CS research in different universities. We can see two highest peaks from Toronto and Waterloo. It is well-known that the two universities have the largest Computer Science programs in Ontario. The next four universities by the number of CS webpages are York, Ottawa, McMaster and Western. As far as we know, these universities have an intermediate size of CS departments or CSE (Computer Science and Engineering) departments.

Readers may notice a large number of webpages categorized as "General computer science" (i.e., not predicted into any specific subtopics). We find that these webpages are either about general topics of CS, such as home page, research interest list and program introduction, or related CS pages from the other departments, such as *CHASS Facilities* (in Faculty of Arts of Toronto)[19], *Arts Computing Office* (in Faculty of Arts of Waterloo) [20] and *Social Science Network and Data Services* (in Faculty of Social Science of Western) [21]. Due to the large number of CS related webpages in other disciplines, the webpages classified into "General computer science" are far more than the other subtopics.

[19]`http://www.chass.utoronto.ca/facilities/`
[20]`https://artsonline.uwaterloo.ca/aco/`
[21]`http://ssnds.uwo.ca/rooms.asp`

To summarize, in this subsection, we conduct comprehensive analysis of the academic topics mined by our hierarchical classification approach. We find that the results are reasonable and consistent with the common sense of topic distribution in universities. In the next section, we will conduct a comparative experiment to demonstrate the superiority of our approach to a state-of-the-art topic modeling approach.

## 6.6 Comparing Results from Topic Modeling Approach

In this section, we compare the academic topics mined by our hierarchical classification framework to the results mined by a state-of-the-art topic modeling approach, namely the LDA (Latent Dirichlet Allocation) model (see detailed review in Chapter 2.4).

We use the popular Mallet [84] package for learning the LDA model from Western. We set the number of topics as 200, a common topic parameter for learning a LDA model. The hyperparameters alpha and beta are set to 0.25 (50/200) and 0.1 according to [46].

### Analyzing Topic Structure

We first analyze the mined topic structure. As an unsupervised learning algorithm, the LDA model does not need training data. In a LDA model, each topic is learned as a word distribution directly from the testing dataset. The most probable words (with largest probabilities) can be used to interpret the semantics of topics. We tabulate the top ten most probable words from some sample topics in table 6.4.

Unfortunately, we see that the topics mined by the LDA model are very noisy. Although the LDA model can discover some academic topics, such as T1 (”Programming Language”), T26 (“Astronomy”), T37 (“Chemistry”) and T49 (“Biology”), we can find many non-academic and even unmeaningful topics, such as T11 (“Assignment”), T22 (“Mustang”) and T39 (“Building”). This is due to the reason that the LDA learning algorithm (e.g., the Gibbs sampling [46]), as an unsupervised model, does not have explicit distinction between academic topics and non-academic topics. All the topics in the LDA learning procedure are just word distribution estimated from the co-occurrence of word-document data.

On the other hand, in our hierarchical classification based approach, the mined topics (see Table 6.6 and Table 6.7) are more meaningful and controllable. This is because our hierarchical classification system is guided by a manually-built academic topic hierarchy and trained on a large number of labeled webpages. This is one of the major advantages of supervised classification approaches over unsupervised topic modeling approaches.

In addition, it should be noted that the labels (names) of those topics in LDA have to be labeled by a human. This could be very costly for a real world application where automated data mining process is crucial. While in our hierarchical classification approach, the topic hierarchy is built off-line.

### Analyzing Topic Distribution

Secondly, we analyze the quality of the inferred topic distribution of each webpage. Following the same experimental methodology in Section 6.5.2, we also focus on the topic prediction

Table 6.4: Some sample topics mined by the LDA model from webpages in Western. Academic topics are marked in bold.

| Topic | Top ten most probable words |
|---|---|
| **T1** | **class, methods, public, java, int, string, return, objectives, lang, field** |
| T2 | award, scholarship, president, canada, faculty, year, ontario, graduate, honour, communications |
| T3 | vol, iss, papers, science, discussion, home, issue, search, psc, series |
| **T4** | **canadian, poetry, poems, early, love, long, poets, thy, day, heart** |
| T5 | chrw, comments, radio, chrwradio, events, view, music, dolor, year, london |
| T6 | oset, spline, components, change, radius, rgb, create, colour, color, declared |
| T7 | post, cpsx, events, news, mission, science, comments, space, march, read |
| T8 | chrw, comments, london, radio, blogs, week, chrwradio, year, music, news |
| T9 | papers, present, research, questions, student, discussion, project, topics, work, including |
| T10 | exam, psychology, chapter, student, lecture, test, assignments, final, office, class |
| T11 | assignments, student, exam, read, lecture, class, lab, final, week, mark |
| T12 | teaching, graduate, student, program, learning, faculty, workshops, research, resources, award |
| T13 | computer, student, science, web, research, csd, information, department, grad, graduate |
| **T14** | **music, band, album, song, record, rock, cd, play, sound, london** |
| T15 | war, world, life, time, people, year, history, church, live, god |
| T16 | media, program, information, student, studies, courses, admission, application, mit, faculty |
| T17 | time, people, year, work, don, good, feel, life, student, day |
| T18 | meeting, committee, council, club, members, usc, executive, board, policy, reporter |
| **T19** | **patients, clinical, cancer, treatment, studies, trial, disease, year, risk, group** |
| **T20** | **network, system, services, computer, mail, message, internet, user, information, security** |
| T21 | gazette, student, editor, comments, archive, post, news, read, advertise, contribute |
| T22 | form, men, women, mustang, club, athletics, schedule, information, student, campus |
| **T23** | **music, performance, faculty, piano, wright, orchestra, opera, record, instruments, student** |
| T24 | limited, canada, corporate, library, company, corp, funding, resources, canadian, mining |
| T25 | research, development, canada, ontario, technology, industry, london, city, centre, governance |
| **T26** | **physical, star, astronomy, earth, planet, sun, orbit, energy, magnetic, mass** |
| T27 | scholarship, geography, science, author, search, research, ontario, digital, faq, journal |
| T28 | web, standards, news, advertise, campus, daily, browser, feed, student, services |
| T29 | web, standards, news, advertise, daily, campus, browser, feed, london, student |
| **T30** | **blood, patients, heart, increase, infusion, iv, transfusion, dose, cardiac, effective** |
| T31 | wed, thu, tue, fri, news, gazette, archive, student, links, campus |
| T32 | neurological, dr, science, residence, clinical, london, neuro, view, department, bio |
| **T33** | **pain, surgery, stroke, patients, research, muscle, studies, injury, knee, joint** |
| T34 | wed, thu, tue, fri, gazette, archive, arts, entertainment, links, sports |
| T35 | studies, history, culture, literature, religious, anthropology, compared, french, topics, theology |
| T36 | meteor, meteorite, research, public, project, video, overview, photos, web, camera |
| **T37** | **chemistry, reaction, protein, structure, mass, sample, acid, nmr, solution, chemical** |
| **T38** | **language, speech, hear, science, research, pathology, communications, disorders, children, audiology** |
| T39 | room, turn, door, open, north, east, building, day, road, time |
| **T40** | **patients, health, care, program, research, cancer, family, london, lhsc, hospital** |
| T41 | english, literature, book, history, read, century, text, ed, author, studies |
| T42 | writing, assignments, class, student, chapter, grade, due, week, essay, medical |
| T43 | web, standards, news, advertise, campus, daily, browser, feed, london, services |
| T44 | web, standards, research, graduate, student, browser, science, health, information, page |
| T45 | gi, nc, tx, synthase, dehydrogenase, ref, phosphate, sp, protein, subunit |
| T46 | click, map, number, selected, line, data, file, true, enter, list |
| T47 | ontario, research, canada, professor, association, dr, conference, toronto, department, canadian |
| **T48** | **health, risk, diabetes, women, vaccine, test, hiv, clinical, pregnancy, medical** |
| **T49** | **biology, plant, animal, species, ecology, evolution, change, nature, bird, human** |
| T50 | year, work, people, time, london, life, love, family, day, stories |

results in the Faculty of Science of Western. We compute the average topic probabilities for webpages belonging to these departments. As we are not aware of the true semantics of each topic inferred by the LDA model, we use the top five most probable topics for each department. Figure 6.12 plots the heat map matrix. We can see a relatively strong diagonal in the matrix. This means that the LDA model can also work well to infer related topics for each department.

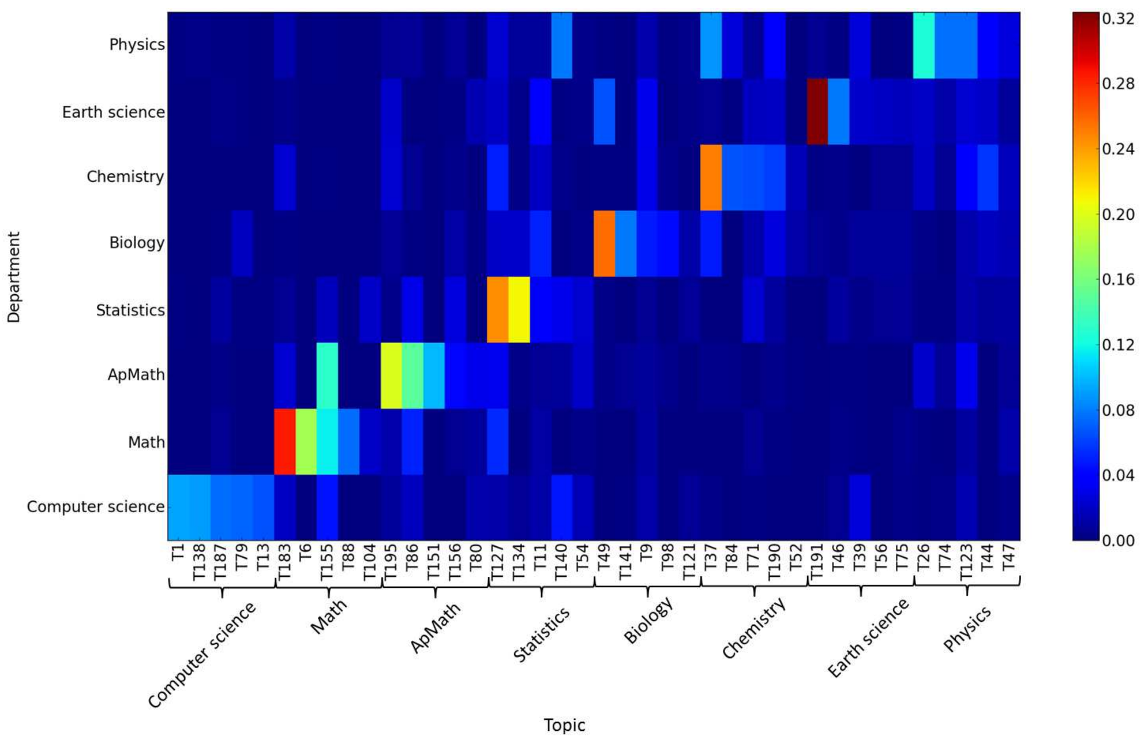


Figure 6.12: A heat map matrix showing the average topic probabilities by the LDA model for webpages belonging to departments in the Faculty of Science of Western. Each row is a vector of topic probabilities.

However, comparing the results by our hierarchical classification approach in Figure 6.10, we can not see the strong correlation among the four closely related departments (CS, Math, ApMath and Statistics). The pattern that CS topics are shared among most of departments is also not observable. Even by trying more topics (e.g., 300 and 400) and repeating the LDA learning , we still can not observe the correlation among these departments. Actually, for department of CS, we find that the five most probable topics by LDA are either about specific topics, such as “Programming language” or non-academic information such as the CS department website. It is unlikely that such topics can be shared by the other departments. On the other hand, our hierarchical classification approach can exploit the topic hierarchy to infer topics in different levels of granularity so that specific CS topics (e.g., “Machine learning” and “Data mining”) related to other research disciplines (e.g., “Statistics”, “Applied mathematics”) can be learned. This is why our approach can detect the topic correlation among related departments.

To summarize, in this experiment, we compare the academic topics mined by our hierarchical classification approach with the LDA topic modeling approach. We find that the results mined by our approach are better than the LDA topic modeling approach.

## 6.7 Summary

In this chapter, we have presented a novel hierarchical classification framework for mining academic topics from webpages in the 12 largest universities in Ontario, Canada. According to our comprehensive experiments, the academic topic pattern mined by our hierarchical classification framework is reasonable, consistent with the common sense of topic distribution in these universities, and better than the traditional LDA topic modeling approach. In the next Chapter, we will integrate the hierarchical classification results into the implementation of SEEU and conduct usability studies to evaluate the usefulness of SEEU.

# Chapter 7

# System Implementation and Usability Studies

In this chapter, we describe the system implementation of SEEU and conduct two usability studies to evaluate SEEU. We first describe the main system components of SEEU. After that, we analyze the university dataset characteristics which are related to the implementation. Finally, we conduct two usability studies to evaluate SEEU and discuss the evaluation results.

The useability studies in Section 7.3.2 were in collaboration with Dr. Charles Ling and Dr. Ali Bou Nassif from the ECE department of Western. This work was included in the submission to the *IEEE Transactions on Knowledge and Data Engineering* (IEEE TKDE) [75].

## 7.1 System Implementation

In this section, we describe the implementation details of SEEU at the system level (i.e. focusing on the functionality and connection between different system modules).

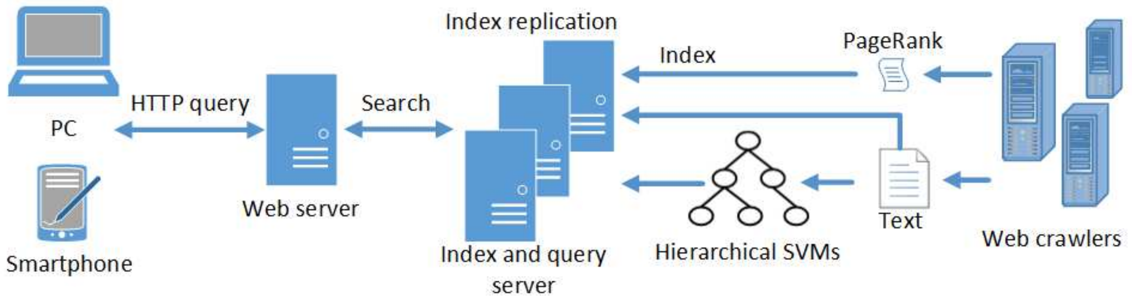


Figure 7.1: The system view of SEEU.

Figure 7.1 presents the system view of the SEEU search engine, which includes the user interface (for both PC and mobile phones), the web server, the indexing and query server, the hierarchical SVMs and the web crawlers. Here, we give a high-level overview of the five system modules from the client side to the server side. At the client side, the user interface visualizes the search results and help users navigate during search sessions. The web server acts as an agent between the user interface and the indexing and query server. It receives user queries, translates them into internal query expressions (to be analyzed by the indexing server), retrieves the ranked results from the indexing server and returns them to users. The indexing and query server is the core of our search engine. It indexes the text data, SVM classification probabilities outputted by the hierarchical SVMs and the PageRank score of each webpage into the search engine, and responses to the queries sent through the web server. The last component of the web crawlers crawl the university webpages and extract the text data and PageRank scores.

As we have already presented the classification and ranking algorithms in Chapter 3, Chapter 4 and Chapter 6, we do not repeat the algorithm of hierarchical SVMs and the method of ranking in hierarchies. In the following subsections, we will briefly discuss the implementation of the other four system modules.

### 7.1.1 User Interface

In SEEU, with the integration of multiple hierarchies, the interaction between users and the search engine is more complex than traditional keywords-based search engines. Besides the (optional) query keywords, the user interface of SEEU should also manage the contextual category information for navigation, and present them to users in an intuitive way. In addition, as we design both PC and mobile versions of SEEU, the user interface should be easily extended to different platforms.

To deal with these design challenges, we adopt the MVC (Model-View-Controller) architecture [66, 69], a software architecture pattern, to separate the complex representation of contextual information from the user's interaction. In SEEU, the MVC architecture can be instantiated as in Figure 7.2.

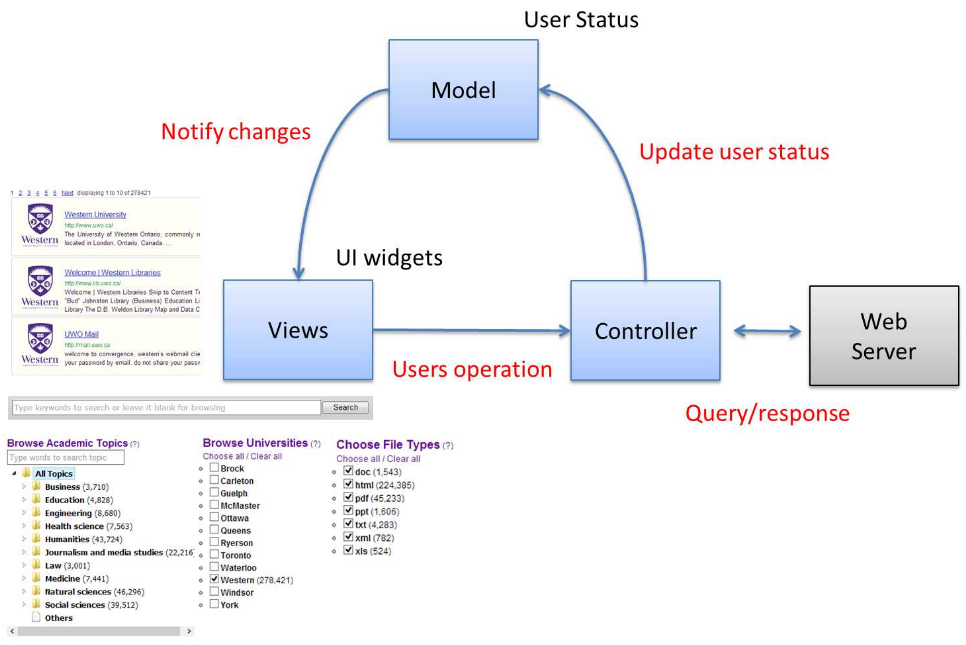
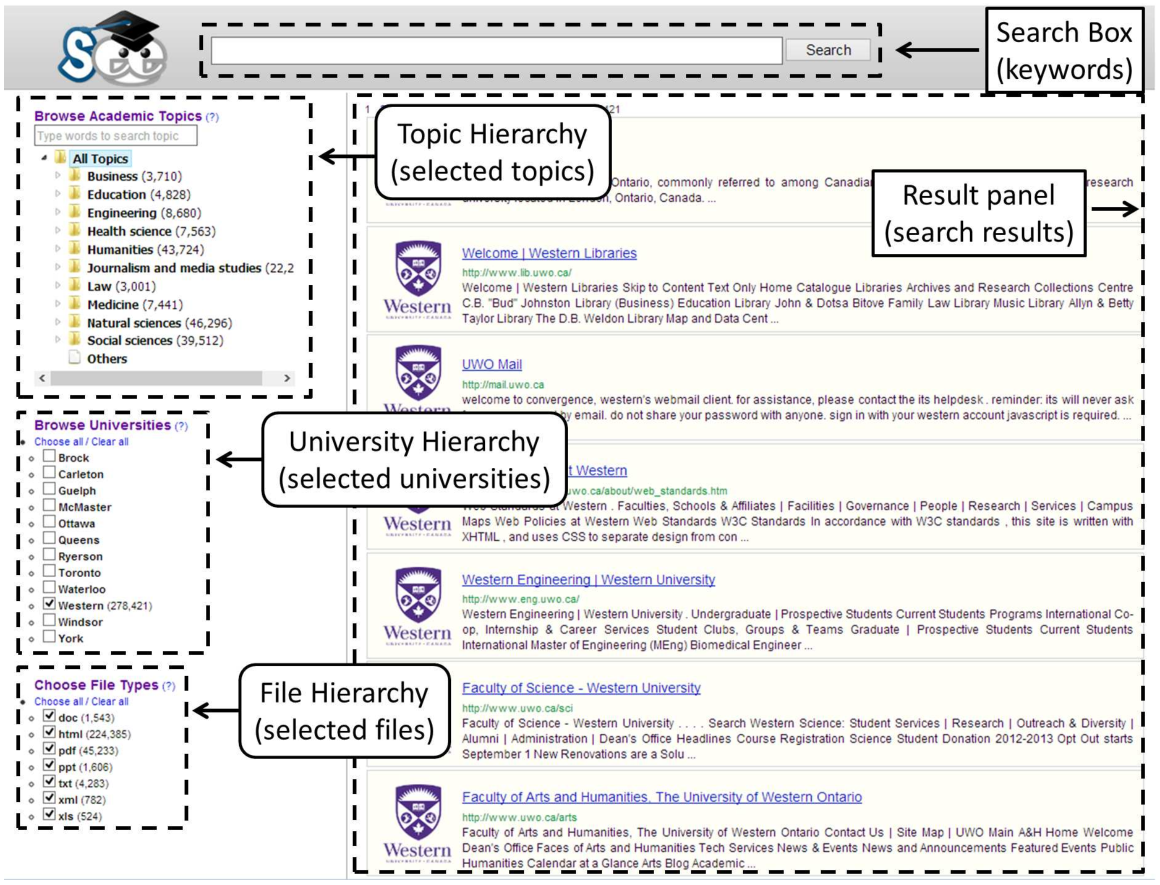


Figure 7.2: The MVC architecture of SEEU user interface. The five UI widgest are attached to the Views.

- The Model manages the current user status in a search session. In SEEU, the user status is a five-tuple array, consisting of the keywords, the search results and three contextual information, i.e., the selected topics, the selected universities and the selected files. To reduce the storage cost of servers, the information of user status is stored at client-side browsers as cookies[1].

- The View mainly deals with the visualization of user status. Each View corresponds to an individual user status. Thus, five different Views are implemented in SEEU, as shown in Figure 7.3. We can see a search box, a topic hierarchy, a university hierarchy, a file type hierarchy and a search result panel.


Search
Search Box (keywords)
Topic Hierarchy (selected topics)
Result panel (search results)
University Hierarchy (selected universities)
File Hierarchy (selected files)

Browse Academic Topics (?)
Type words to search topic
All Topics
Business (3,710)
Education (4,828)
Engineering (8,680)
Health science (7,563)
Humanities (43,724)
Journalism and media studies (22,2
Law (3,001)
Medicine (7,441)
Natural sciences (46,296)
Social sciences (39,512)
Others

Browse Universities (?)
Choose all / Clear all
☐ Brock
☐ Carleton
☐ Guelph
☐ McMaster
☐ Ottawa
☐ Queens
☐ Ryerson
☐ Toronto
☐ Waterloo
☑ Western (278,421)
☐ Windsor
☐ York

Choose File Types (?)
Choose all / Clear all
☑ doc (1,543)
☑ html (224,385)
☑ pdf (45,233)
☑ ppt (1,606)
☑ txt (4,283)
☑ xml (782)
☑ xls (524)

1 21

Ontario, commonly referred to among Canadia research
, Ontario, Canada. ...

Welcome | Western Libraries
http://www.lib.uwo.ca/
Welcome | Western Libraries Skip to Content Text Only Home Catalogue Libraries Archives and Research Collections Centre C.B. "Bud" Johnston Library (Business) Education Library John & Dotsa Bitove Family Law Library Music Library Allyn & Betty Taylor Library The D.B. Weldon Library Map and Data Cent ...

UWO Mail
http://mail.uwo.ca
welcome to convergence, western's webmail client. for assistance, please contact the its helpdesk . reminder: its will never ask by email. do not share your password with anyone. sign in with your western account javascript is required. ...

t Western
uwo.ca/about/web_standards.htm
Western . Faculties, Schools & Affiliates | Facilities | Governance | People | Research | Services | Campus Maps Web Policies at Western Web Standards W3C Standards In accordance with W3C standards , this site is written with XHTML , and uses CSS to separate design from con ...

Western Engineering | Western University
http://www.eng.uwo.ca/
Western Engineering | Western University . Undergraduate | Prospective Students Current Students Programs International Co-op, Internship & Career Services Student Clubs, Groups & Teams Graduate | Prospective Students Current Students International Master of Engineering (MEng) Biomedical Engineer ...

Faculty of Science - Western University
http://www.uwo.ca/sci
Faculty of Science - Western University . . . . Search Western Science: Student Services | Research | Outreach & Diversity | Alumni | Administration | Dean's Office Headlines Course Registration Science Student Donation 2012-2013 Opt Out starts September 1 New Renovations are a Solu ...

Faculty of Arts and Humanities, The University of Western Ontario
http://www.uwo.ca/arts
Faculty of Arts and Humanities, The University of Western Ontario Contact Us | Site Map | UWO Main A&H Home Welcome Dean's Office Faces of Arts and Humanities Tech Services News & Events News and Announcements Featured Events Public Humanities Calendar at a Glance Arts Blog Academic ...

Figure 7.3: The implementation of Views in SEEU. Each View corresponds to an individual user status (wrapped by brackets).

- The Controller acts as a bridge between the Model and the five Views. On one hand, when the Model changes, it sends *update* commands to all Views to change their presentation of the related user status. On the other hand, when the controller receives user actions from Views, it queries the web server (see later in Section 7.1.2) to retrieve ranked results and change the Model data.

[1]See discuss of web browser cookie in `http://en.wikipedia.org/wiki/HTTP_cookie`.

The most complex UI widget (View) in SEEU is the topic hierarchy as shown in Figure 7.4. The basic functionality of the topic hierarchy is to help users explore the indexed webpages and be effectively navigated during the search session. For example, when a user clicks a topic category, such as “Natural sciences”, the topic hierarchy shows not only its child subcategories but also its sibling categories. In addition, no matter what keywords the user types, the currently selected category is always be highlighted.

Readers can see numbers surrounded by brackets besides each category. These numbers denote the number of webpages classified into each category of the hierarchy under the conditions that they also match the current keywords and satisfy the constrains in other hierarchies (i.e., the selected universities and the selected file types).[2] Similar to the faceted search [51], users can use these numbers as a guide to explore the topic hierarchy, such as skipping smaller categories or rolling up or drilling down to different categories. With these rich contextual information, we believe that user’s search experience can be greatly improved.

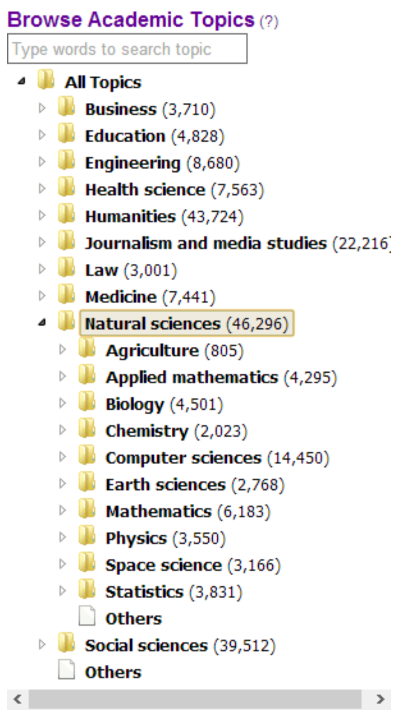

Browse Academic Topics (?)
Type words to search topic
- All Topics
  - Business (3,710)
  - Education (4,828)
  - Engineering (8,680)
  - Health science (7,563)
  - Humanities (43,724)
  - Journalism and media studies (22,216
  - Law (3,001)
  - Medicine (7,441)
  - Natural sciences (46,296)
    - Agriculture (805)
    - Applied mathematics (4,295)
    - Biology (4,501)
    - Chemistry (2,023)
    - Computer sciences (14,450)
    - Earth sciences (2,768)
    - Mathematics (6,183)
    - Physics (3,550)
    - Space science (3,166)
    - Statistics (3,831)
    - Others
  - Social sciences (39,512)
  - Others

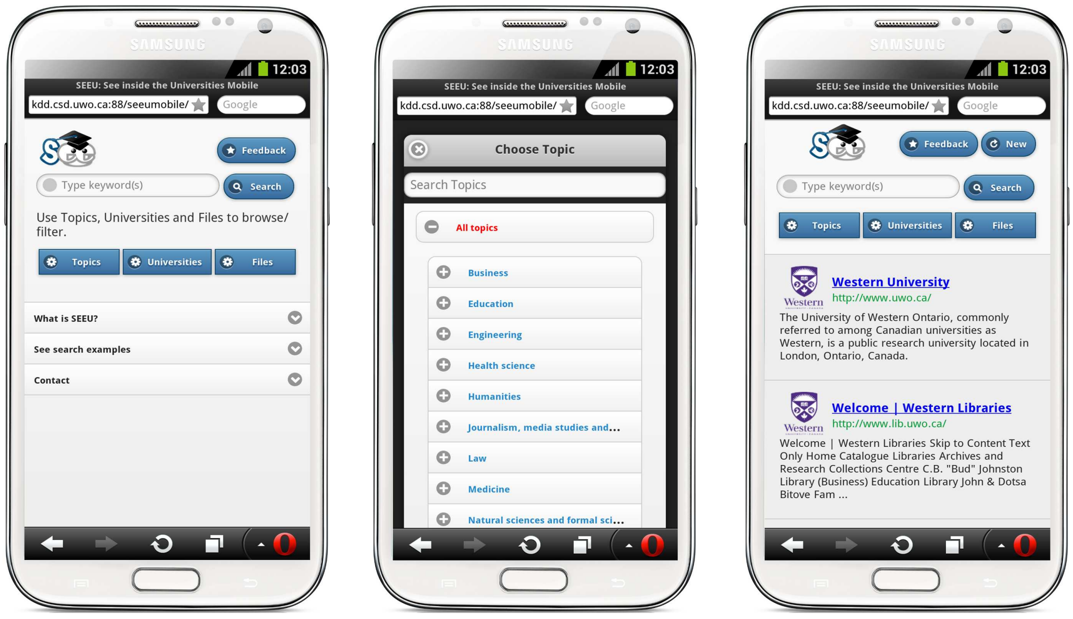
Figure 7.4: The topic hierarchy in SEEU. A user clicks the “Natural sciences” category.

A novel feature of SEEU is that these numbers are automatically updated according to the different keywords (or categories in other hierarchies) that users issue. Readers may ask how do we efficiently compute these numbers on the client side? Actually, these numbers are acquired by sending a special search request to the indexing and query server that only returns the number of matched webpages rather than a list of indexed webpage features (i.e., URL, title, description, keywords, and so on). Thus, the network transmission cost can be greatly saved (just a list of integers). This technique is also used in the other two hierarchies.

In SEEU, the MVC architecture is implemented as a *jQuery/Ajax*[3] framework on the client-side. Although we only discuss the UI implementation of SEEU for the standard PC version, this framework can be easily adapted to mobile platforms. We only need to change the visualization code of the different Views. Figure 7.5 presents the mobile user interface. Due to the limited screen size, the three hierarchies (Views) are replaced by three buttons on the home page (see Figure 7.5(a)). When users click one of the buttons (say Topics), SEEU Mobile opens a popup dialog (of full screen size) that shows the hierarchy similar to the PC version (see Figure 7.5(b)). Users can click a desired subcategory to browse or filter the search results (see Figure 7.5(c)).

[2]The empty categories (i.e., no results) are hidden by SEEU. This is to avoid user confusion when they click an empty category but do not see any results.

[3]http://jquery.com/



The University of Western Ontario, commonly referred to among Canadian universities as Western, is a public research university located in London, Ontario, Canada.

Welcome | Western Libraries Skip to Content Text Only Home Catalogue Libraries Archives and Research Collections Centre C.B. "Bud" Johnston Library (Business) Education Library John & Dotsa Bitove Fam ...

(a) Home page (b) Topic hierarchy dialog (c) Result page

Figure 7.5: The home page, the topic hierarchy dialog and the result page in SEEU Mobile. The university dialog and the file dialog are similar to the topic dialog.

### 7.1.2 Web Server

The web server in SEEU mainly acts as an agent between the user interface and the indexing and query server. It translates user queries (i.e., user status sent from the controller in MVC implementation) into internal query expressions (explained later in Section 7.1.3), retrieves the ranked results from the indexing and query server and returns them to users.

It may be argued that it is not necessary to add this intermediate layer as we can directly expose the internal query API to users. However, we believe that it is worth to do so. Because we can hide the implementation details of the indexing and query server from the user interface. If we change the query API, we do not need to change any code on the client side (e.g., both PC and mobile UI). Moreover, adding an intermediate layer can also ensure the security of SEEU. It should be noted that we also provide APIs to modify or delete indexed data. Exposing such an API to users could make SEEU vulnerable.

In SEEU, the web server also acts as a load balancing server. When users search documents, the main computation is performed at the indexing and query server. When many users visit SEEU simultaneously, only using one indexing server may seriously increase response time. To solve the bottleneck of querying index server, we use the *replication* technique to make several copies of the index server to reduce response time. At the web server sides, we implement a simple *Round-robin* strategy for load balancing. Specifically, given $m$ index servers with process IDs as 0, 1, ..., $m-1$, the web server dispatches the $n$th user query to the machine with ID=$n$ Mod $m$ where Mod is the modulo operation. In ideal scenarios, the throughput rate of SEEU can be increased by $m$ times.

### 7.1.3 Indexing and Query Server

We use *Apache Solr*[4], a popular open source search platform, as the main indexing and query server. The major features of *Solr* include powerful full-text search, hit highlighting, near real-time indexing, faceted search, and so on. As the indexing and query server is the core of SEEU search engine, we briefly describe its implementation in this subsection.

To index documents into *Solr*, we need to customize the *Solr* configuration file that defines the document attributes to be indexed. In SEEU, we define the *Solr* configuration file to contain a unified document ID, a page importance score, multiple text features and a list of probability values for each category in SEEU's topic hierarchy. Table 7.1 tabulates a brief summary of these attributes. Most of these attribute values are derived from the subsystems we presented in previous chapters. Specifically, the PageImportance attribute (2) is derived from the importance score generated by crawlers (see Appendix A); the text attributes (3 to 8) are extracted by the MapReduce feature extraction tool (see in Section 3.1); the file type attribute (10) and the university attribute (11) can be easily extracted from URLs; the CatProb attributes (12-475) are outputted by our hierarchical SVM classifiers (see Section 6.5). These attributes are used to compute the ranking score for each indexed document during query.

Table 7.1: The indexed document attributes in *Solr*.

| NO. | Name | Description |
|---|---|---|
| 1 | ID | Unique document identifier |
| 2 | PageImportance | Page importance score |
| 3 | URL | Page URL |
| 4 | Title | Page title |
| 5 | Description | Page description (empty for non HTML pages) |
| 6 | Keywords | Page keywords (empty for non HTML pages) |
| 7 | Content | Page content |
| 8 | AnchorText | Anchor text from inbound links |
| 10 | FileType | Page file type |
| 11 | University | The name of university where the page is crawled |
| 12-475 | CatProb(ID) | SVM probability of a category |

To query documents in *Solr*, we need to implement our ranking function in *Solr*. Recall that we use a linear function $f(\vec{\mathbf{x}}) = \vec{\mathbf{w}} \cdot \vec{\mathbf{x}}$ to output a ranking score for each indexed document (see Chapter 4). In *Solr*, a standard way to search and rank documents is to use the HTTP/XML API. Our linear ranking function can be implemented by specifying *Solr*'s popular Extended DisMax (eDisMax) query parser in HTTP requests. For example, to search for "data mining" in "Natural science" → "Computer sciences" in *Solr*, we can issue a HTTP request to *Solr* with parameters:

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

| | | |
|---|---|---|
| defType | = | *edismax* |
| q | = | *data mining* |
| qf | = | *URLˆ1.15 Titleˆ2.5 Descriptionˆ3.2 Keywordsˆ3.5 Contentˆ1.1 AnchorTextˆ3.0* |
| tie | = | 1 |
| bf | = | *sum(product(PageImportance,2.2),product(CatProb403,CatProb456,1.13))* |

We list the meaning of the eDisMax request parameters as follows

- *defType* specifies the type of query. In SEEU, we use the eDisMax query parser.
- *q* specifies the query terms, such as "data mining". We can use *q=*:** to issue empty-word queries (i.e., browsing).
- *qf* specifies the text attributes used in our ranking function. The superscripts denote the weights of attributes. For example, *Titleˆ2.5* means the weight of *Title* score is 2.5.
- *tie* specifies the ranking score aggregation method on the *qf* attributes. In *Solr*, *tie=1* means a linear weighted sum of *qf* parameter values (i.e., $\sum_{f \in d} w(f) \cdot s(f)$ where $w(f)$ denotes the weight and $s(f)$ denotes the score of attribute $f$) while *tie=0* means a maximum function (i.e., $max_{f \in d} w(f) \cdot s(f)$). We use *tie* = 1 in SEEU.
- *bf* specifies the additional boosting scores from non-text attributes (i.e., page importance and category probabilities). In the above example, *sum* and *product* are mathematical functions. *CatProb403* denotes the probability of "Natural science". *CatProb456* denotes the probability of "Computer sciences". The score calculated by the mathematical expression will be added into the final ranking score.

Thus, for the above HTTP request example, the final ranking score for each document in *Solr* is

$$\begin{aligned} Score &= 1.15 \times Score(URL) + 2.5 \times Score(Title) + 3.2 \times Score(Description) \\ &+ 3.5 \times Score(Keywords) + 1.1 \times Score(Content) + 3.0 \times Score(AnchorText) \\ &+ 2.2 \times PageImportance + 1.13 \times CatProb\_403 \times CatProb\_456 \end{aligned}$$

where each $Score$ function is a TF·IDF score between the query and a document attribute. We can see that this ranking score equation is similar to the linear ranking function we discussed in Chapter 4. The weights of the ranking function are stored in our web server after the training of the ranking system.

We discuss the implementation of hierarchical refinement (selecting categories to refine results) in SEEU. Besides these basic parameters, *Solr* provides a filter parameter $fq$ to restrict results by constrain. Developers can add any number of $fq$ parameters in a HTTP request to refine the results. In SEEU, we can use $fq$ parameters to refine the results in three hierarchies (i.e., topics, file types and universities). For the previous example, we can

- use fq=CatProb403:[0.3 TO 1]&fq=CatProb456:[0.6 TO 1] to restrict the results in the category path of "Natural science" (CatProb403) → "Computer sciences" (CatProb456). The meaning of CatProb403:[0.3 TO 1] is that the probability of documents belonging to "Natural science" must be between 0.3 and 1. The number 0.3 is actually the prediction threshold tuned by our hierarchical SVM classifiers (see Section 6.4.2).

- use fq=FileType:(PDF OR DOC) to restrict the file types in “PDF” or “DOC”.

- use fq=University:(Western OR Toronto) to restrict the results only in Western or Toronto.

All of the parameters composition and transformation to HTTP requests are conducted by the Controller in the MVC implementation.

### 7.1.4 Web Crawlers

We use the popular *Apache Nutch* crawler[5] in SEEU. *Nutch* is an open source web crawler written in Java. It has a highly modular architecture. Many useful plug-ins, such as data retrieval, hyperlink analysis and document parsing, are already integrated into *Nutch*. Thus, we can collect webpages in an automated manner, and reduce lots of implementation work. The detailed configuration of *Nutch* can be found in Appendix A.

In this section, we have discussed the implementation of SEEU in a high level. We discuss the challenges of building SEEU, and propose several effective implementation strategies to solve them. In the next section, we analyze the characteristic of the SEEU dataset which is closely related to the implementation of SEEU search engine.

## 7.2 Characteristics of University Webpage Data

By using *Nutch*, we start 12 independent crawling jobs and collect 1,974,172 webpages from the 12 universities. In this subsection, We analyze the characteristics of the SEEU dataset, which include the page importance scores, the file type distribution and the web graph generated by *Nutch*.

Firstly, we examine the page importance scores in the SEEU dataset. We normalize the page importance scores generated by *Nutch* into the range [1, 100], and plot their histogram in ten bins in Figure 7.6. We can see a skewed page number distribution. That is, most webpages have very low importance scores. Specifically, about 99% of the webpages have importance score less than 10. It means that most of the webpages in universities have few inbound links. This is reasonable because it is not common to see hyperlinks between different departments. For the 1% of webpages with high importance scores, we find that they are about university home pages, department home pages or student service related pages (i.e., library, admission). This may be due to the reason that most universities deploy unified HTML templates that contain important entry pages (e.g., university home page or department home page) for the web design. Thus, those primary entry pages appear in most of the webpages (embedded in the same template). With a large number of inbound links, their importance scores are very high.

[5]The home page of *Nutch* is `http://nutch.apache.org/`.

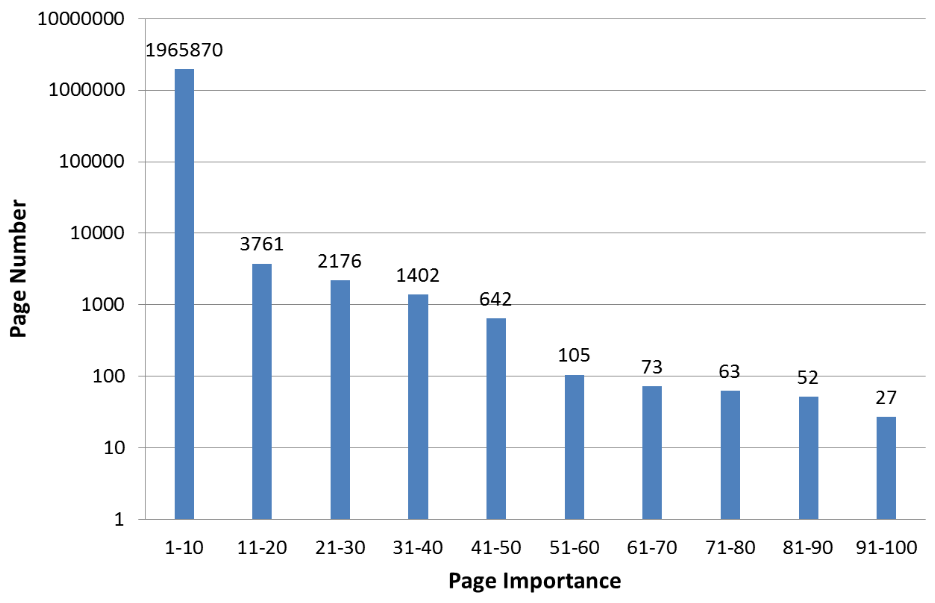


Figure 7.6: Histogram of the page importance scores in SEEU dataset.

Secondly, we study the file type distribution in our data collection (see Figure 7.7). As the majority of university documents are published as HTML pages, it is not surprising to see that the HTML pages comprise over 80% of the dataset in our collection. The second largest file type is the PDF file. This is because people in universities usually publish research papers and course materials as PDF files. For the remaining five percent of the data collection, they are *TXT*, *DOC*, *PPT*, *XML* and *XLS*. It should be noted that each file type does not mean a single file format. For example, the *DOC* category includes both Word 2003 (.doc) and Word 2007 (.docx) files.

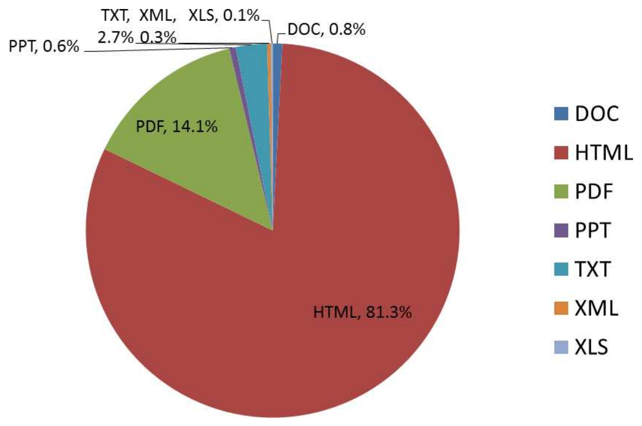


Figure 7.7: The file type distribution of crawled webpages in SEEU dataset.

Thirdly, we study the hyperlink graph (or web grpah) generated by *Nutch*. We do not study the properties of hyperlinks inside each university but focus on the relations among the 12 universities by analyzing the inter-university hyperlinks. For each university, we compute the proportions of hyperlinks pointing to each of the other universities. For example, for Western, the number of hyperlinks and corresponding proportions (in brackets) to other universities is tabulated below.

| | | | |
|---|---|---|---|
| Brock: 197 (0.03) | Carleton: 137 (0.02) | Guelph: 267 (0.04) | McMaster: 625 (0.08) |
| Ottawa: 276 (0.04) | Queen's: 604 (0.08) | Ryerson: 169 (0.02) | Toronto: 2219 (0.3) |
| Waterloo: 720 (0.09) | Windsor: 1731 (0.23) | York: 634 (0.08) | |

We plot the results as a heat map matrix in Figure 7.8.[6] Each row represents the hyperlink proportion vector for a university. We can find three types of university relations from this figure.

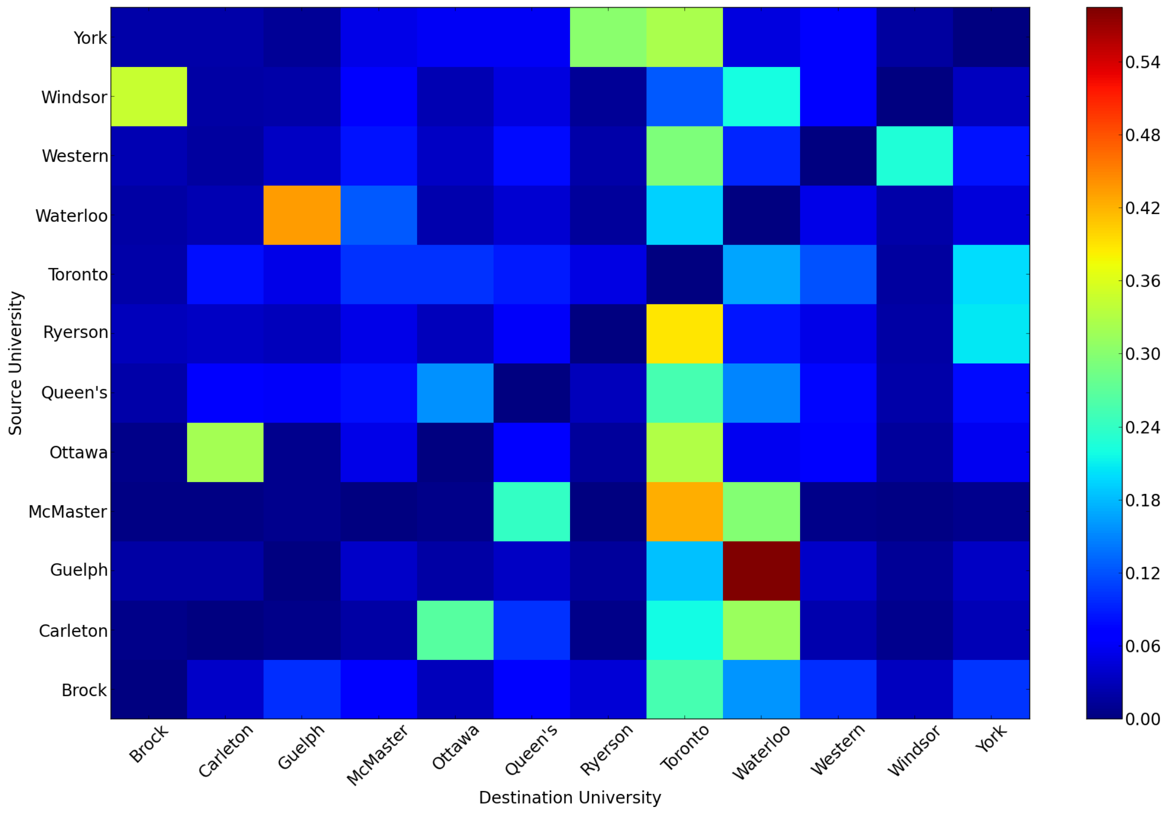


Figure 7.8: The proportions of hyperlinks pointing to each of the other universities in SEEU dataset. Each row is a vector of hyperlink proportion.

- Relations by authority. We can see that nearly all the universities have a relatively strong connection to Toronto. By analyzing the detailed hyperlink URLs, we find that most of the hyperlinks are pointing to the UT Library (`www.library.utoronto.ca`). This could be due to the reason that UT Library is the largest in Ontario so that the references from other universities are high.

- Relations by regions. Several strong relations between pairs of universities are based on regional neighborhood, such as Ottawa-Carleton, Guelph-Waterloo, and Ryerson-Toronto-York. For example, the two strong cells between Guelph and Waterloo are

[6]The hyperlinks inside each university are extremely larger than the hyperlinks to other universities. For example, the number of hyperlinks insides Western is 17,757,956. We do not show them in this figure.

caused by the *TriUniversity Group of Libraries*[7] where long-term library collaboration exists between the two geographically closer universities.

- Relations by collaboration. Education or research collaboration can also form relations between universities, such as Western-Windsor, McMaster-Queen's, McMaster-Waterloo and Carleton-Waterloo. For example, the impressive number of hyperlinks from Western to Windsor is caused by the program expansion of Schulich Medicine School of Western to Windsor; the hyperlinks from McMaster and Queen's are made by the project of *Historical Perspectives on Canadian Publishing*[8] collaborated by McMaster and Queen's; the relations between Carleton and Waterloo is formed by the hyperlinks from *Education Development Centre* in Carleton to *Centre for Teaching Excellence Blog* in Waterloo.

Such relations are revealed from the traditional link analysis. However, the web links between universities are actually very sparse compared to the total number of webpages. For example, only 0.4% links in the SEEU dataset have connection to Western. Only using link analysis may not reflect the true collaboration and be difficult to reveal the potential collaboration between universities. This can again be seen as a motivation examples for the SEEU search engine which relies on content based data mining method (i.e., keywords matching and text classification) for facilitating research collaboration between universities.

In this section, we have analyzed the characteristic of SEEU dataset. The analysis poses a strong motivation of SEEU search engine for facilitating research collaboration between universities. In the next section, we evaluate SEEU by conducting two usability studies in our university.

## 7.3 Usability Studies

We conducted two usability studies to evaluate SEEU. One was conducted in the Electrical and Computer Engineering Department of our university, and the other one was conducted in the Computer Science Department at the same university.

### 7.3.1 Usability Study in the ECE Department

We collaborated with the ECE (Electrical and Computer Engineering) Department to evaluate the usability of SEEU. The usability study was conducted as an assignment in an undergraduate Software Engineering course[9] in October 2012.

27 undergraduate students in this course were evenly split into three groups (i.e., each with nine people). Before the usability study, the students were asked to review our IJCAI [67] paper to understand the motivation and basic functionality of SEEU. After that, we handed out the assignment which describes the evaluation guidelines including:

[7]`http://trellis3.tug-libraries.on.ca/`

[8]`http://hpcanpub.mcmaster.ca/`

[9]The course name is SE4452a Software Verification and Validation. More information can be found at `http://www.eng.uwo.ca/electrical/education/undergraduate_programs/SE4452A-2012-13Approved.pdf`.

1. Query relevance. Evaluate the result relevance of SEEU (using hierarchy) compared with general search engines (e.g., Google, Bing and Yahoo!) using “site:uwo.ca”.

2. Search functionality. Evaluate the functionality of SEEU, such as browsing in topic hierarchy, searching among multiple universities and filtering by file types.

3. UI adaptability. Evaluate SEEU (compared with general search engines) on different platforms that include but not limited to desktop PCs, MacBooks, and smart phones.

Students were asked to perform many test cases based on but not limited to the assignment guidelines we gave to them. For each test case, students reported either it is accepted or rejected. Specifically, an accepted case means that the search quality or UI adaptability of SEEU is better or comparable to the general web search engines. Finally, each group submited a professional report including all the test cases that they did.

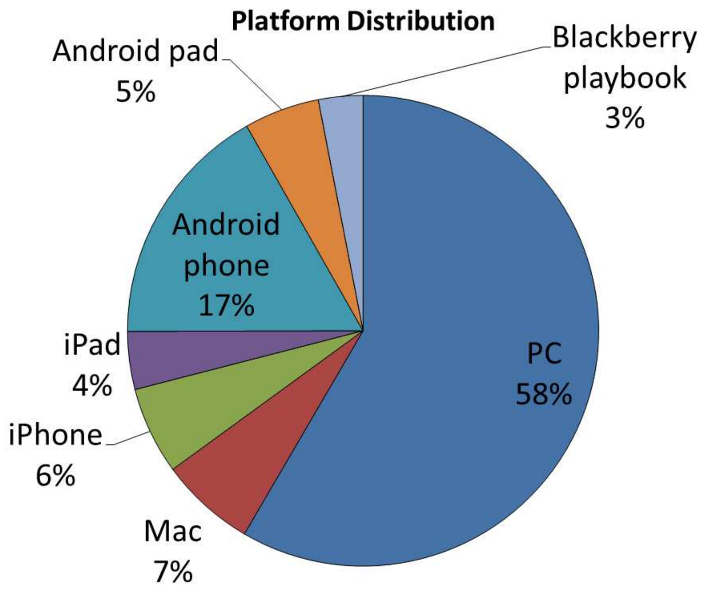
We received the test reports in early November 2012. In total, 1,082 test cases were conducted by students. We tabulate the results in Table 7.2. As we can see, in total, about 77% of test cases are accepted by students. For each specific evaluation task, the result is also very good. For example, the accept rate of query relevance is above 70% which is quite good for an initial design of SEEU compared with commercial search engines (i.e., *Google Search Western*). This result demonstrates that in university search domain, combing keywords-based search with hierarchies (e.g., topic, universities and file types) can improve search result relevance.

Table 7.2: The results of a usability study in the ECE department. Accept means the search quality or UI usability of SEEU is better or comparable to the general web search engines with “site:uwo.ca”.

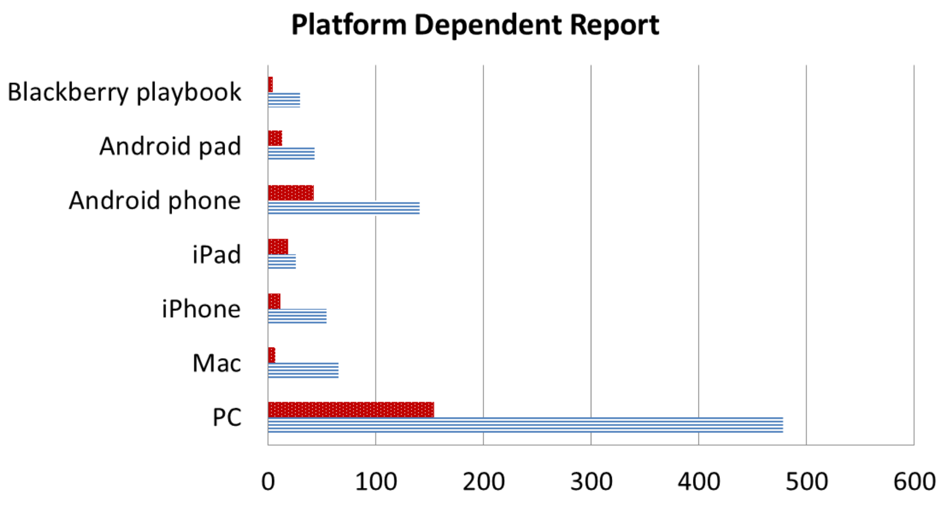
| Content | Accept | Reject | Total | Accept Rate |
|---|---|---|---|---|
| Query relevance | 206 | 87 | 293 | 70.31% |
| Search functionality | 331 | 88 | 419 | 79.00% |
| UI adaptability | 297 | 73 | 370 | 80.27% |
| Total | 834 | 248 | 1,082 | 77.08% |

The highest accept rate is for the UI adaptability. From Figure 7.9(a), we find that students have evaluated SEEU on seven platforms including *PC*, *Mac*, *iPhone*, *iPad*, *Android Phone*, *Android Pad* and *Blackberry Playbook*. We can see that 35% of test cases were conducted on mobile devices. The overall accept rate on mobile devices (calculated from Figure 7.9(b)) is about 76%[10]. It means that SEEU on mobile devices offers students a better search experience than general web search engines. This is as we expected. In SEEU Mobile, users can click different categories in hierarchies to browse or filter results. This is much easier than typing keywords on small keyboards.

[10]Computed as (54 + 25 + 140 + 43 + 29)/(65 + 43 + 182 + 56 + 33) = 291/379 = 0.7678.



(a) Platform Distribution



| | PC | Mac | iPhone | iPad | Android phone | Android pad | Blackberry playbook |
|---|---|---|---|---|---|---|---|
| 🟥 Reject | 154 | 6 | 11 | 18 | 42 | 13 | 4 |
| ≡ Accept | 478 | 65 | 54 | 25 | 140 | 43 | 29 |

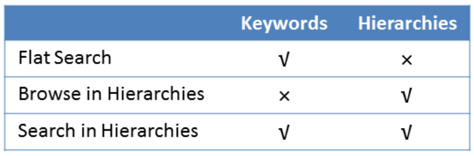
(b) Platform Dependent Report

Figure 7.9: The platform distribution and the detailed platform dependent test report from the usability study in the ECE department. 35% of platforms are mobile devices and the accept rate on mobile devices is 76.78%.

### 7.3.2 Usability Study in the CS Department

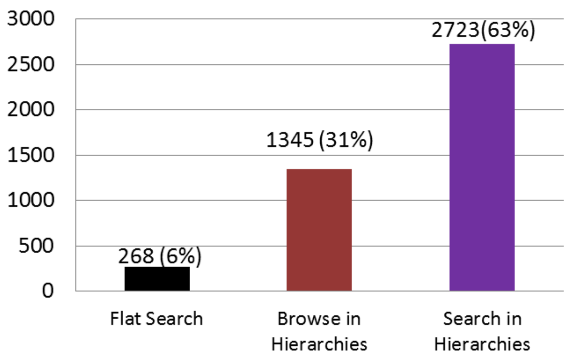
To make a real-world trail of SEEU, we invite all the faculty members and students of the Computer Science (CS) Department to try SEEU. This usability study started from November 2012 till March 2013. We also found many successful search stories. For example, one faculty member was looking for collaborators doing research in "Computer Security". He used SEEU to quickly find researchers in other universities while using *Google Search Western* always returns undesired results. In SEEU, this search task can be easily done by searching "professor" in "Natural Sciences"→"Computer Science"→"Computer Security" with university hierarchies selected.

To study how faculty members and students use SEEU, we further analyze the usage patterns of users. From our search log, there are in total 4,336 usage records[11]. Based on different combinations of using keywords and hierarchies, we categorize these records into three groups as shown in Figure 7.10(a). They are "Flat Search" (no hierarchies), "Browse in Hierarchies" (no keywords) and "Search in Hierarchies". We plot the detailed statistics of the three usage patterns in Figure 7.10(b).



| | Keywords | Hierarchies |
|---|---|---|
| Flat Search | √ | × |
| Browse in Hierarchies | × | √ |
| Search in Hierarchies | √ | √ |

(a) Classification of Usage Patterns



(b) Usage Patterns Distribution

Figure 7.10: The classification of usage patterns and their distribution from the usability study in the CS department.

We can see that the frequency of using SEEU's hierarchies (topics, universities and files) is about 94% while using SEEU without hierarchies is only about 6% percentage. The high traffic of using hierarchies may be due to the easy and fast browsing. When faculty members and students search webpages related to academic topics, clicking the related categories in hierarchies makes SEEU immediately refresh the results for the selected categories. This is much faster than typing keywords in the search box. In addition, among the usage of hierarchies, 31% of usage records do not show any keywords. This means that in these records, faculty members and students simply use SEEU to browse different hierarchies. A possible explanation is that when searching documents in academic topics, users may not always know the right keywords. Sometimes, junior researchers (graduate students) even do not know exactly what they are looking for, as they may not be familiar with the domain. In SEEU, they can select

[11]For this analysis, we do not count the usage from our lab. This ensures that we exclude the traffic caused by the development and testing of SEEU in our lab.

an appropriate category and browse the webpages. This is very useful for surveying a research area or browsing an organization.

Besides the usability study results, we also received many comments from faculty members and students. Basically, most of the comments give us positive ratings on the uniqueness of SEEU. We highlight the main points of their comments.

- *"In general, the hierarchy concept was very well received. It is an intuitive and well-structured method of seeking more specific information."*
- *"The navigation of the menus was very smooth and nice to be to use. It made making selections easy for the different options."*
- *"The ability to specify which university in Ontario that you would like search in further adds to the uniqueness, versatility, and usefulness of SEEU."*
- *"The user interface design of SEEU mobile is beautifully done. It is a great looking webpage on all mobile devices, in both landscape and portrait."*
- *"Most results were only a second or so slower than commercial sized search engines. Assuming the bottleneck here is the server, SEEU is definitely fast enough to be scalable and useful enough to possibly replace current university search solutions."*

From the comments of users, we see a great promise for using SEEU in universities.

## 7.4 Summary

In this chapter, we have presented the system implementation of SEEU. We discuss the challenges to build the SEEU search engine and propose several effective implementation solutions. According to two usability studies (in the ECE and the CS departments in our university), SEEU is favored by the majority of participants.

# Chapter 8

# Conclusions and Future Work

In this thesis, we study the problem of building a novel search engine with integrated hierarchies for universities. In this final chapter, we first summarize the contribution of this thesis, then point out important research problems to be solved in future work.

## 8.1 Conclusions

The web is one of the most important information media for people in universities. Web search engines, due to their success on the general web search, have been adopted by most universities for searching webpages in their own domains. Basically, a user sends keywords to the search engine and the search engine returns a flat ranked list of webpages. However, in university search, user queries are usually related to topics (e.g., academics, campus life, media and so on). Simple keyword queries are often insufficient to express topics as keywords. On the other hand, modern E-commerce sites (such as Amazon and eBay) allow users to browse and search products in various hierarchies (such as product category and region of the world). It would be ideal if hierarchical browsing and keyword search can be seamlessly combined for university search engines. The main difficulty is to define a commonly accepted academic topic hierarchy, and automatically classify and rank a massive number of webpages into hierarchies for universities.

In this thesis, we use machine learning and data mining techniques to build a novel hybrid search engine with integrated hierarchies for universities, called SEEU (**S**earch **E**ngine with hi**E**rarchy for **U**niversities).

In Chapter 3, we develop an effective hierarchical webpage classification system, and demonstrate its efficiency and effectiveness on large-scale webpage dataset. More specifically,

- We implement a webpage feature extraction tool based on Hadoop MapReduce to extract text features from large-scale webpage datasets.
- We implement a parallel hierarchical SVM (Support Vector Machine) classifier based on OpenMPI.
- According to our experiments on the well-known ODP dataset, we empirically demonstrate that our hierarchical classification system is very effective and outperforms the traditional flat classification approach significantly.

In Chapter 4, we propose the ERIC (**E**nhanced **R**anking by h**I**erarchical **C**lassification), a novel ranking framework for search engines with hierarchies. More specifically,

- We integrate hierarchical classification probabilities, text-based query relevance and webpage related metrics (such as PageRank) into a learning to rank framework.
- Empirical study on the well-known TREC (Text REtrieval Conference) web search datasets show that our ranking framework with hierarchical classification outperforms the traditional flat keywords-based search methods significantly.

In Chapter 5, we study the problem of improving hierarchical classification when the labeled dataset is limited. By leveraging the top-down tree structure of hierarchy, we propose a new active learning framework for hierarchical text classification. More specifically,

- We discuss the *out-of-domain* problem in the state-of-the-art parallel hierarchical active learning approach.
- We propose to leverage the top-down tree structure to conduct active learning for the hierarchical classification system in a top-down manner.
- From our experiments on benchmark text classification datasets and the ODP dataset, we find that our hierarchical active learning strategy can achieve good classification performance yet save a considerable number of labeling effort compared with the state-of-the-art active learning methods for hierarchical text classification.

In Chapter 6, we present a novel hierarchical classification framework for mining academic topics in universities. More specifically,

- We build an academic topic hierarchy based on the commonly accepted Wikipedia academic disciplines. We propose to use search engines to quickly collect the training data for this new hierarchy.
- We train a hierarchical classifier based on the new hierarchy and apply it to predict two million university webpages in SEEU.
- According to our comprehensive analysis, the academic topic pattern mined by our system is reasonable, consistent with the common sense of topic distribution in most universities, and better than the traditional LDA topic modeling approach.

In Chapter 7, based on the proposed classification and ranking methods, we discuss the system implementation of SEEU and conduct two usability studies to evaluate the usefulness of SEEU. More specifically,

- We index both webpage text features and hierarchical classification probabilities into the *Solr* search platform, and develop a MVC (Model-View-Controller) based user interface to separate complex visualization from user interaction.
- We conduct two usability studies to evaluate SEEU. One was conducted at the Electrical and Computer Engineering Department at our university, and the other one was conducted at Computer Science Department at our university. SEEU has received excellent user feedback in the initial testing and deployment at our university.

To conclude, the main contribution of this thesis is a novel search engine with integrated hierarchies for universities, called SEEU (**S**earch **E**ngine with hi**E**rarchy for **U**niversities). We discuss the challenges toward building SEEU and propose effective machine learning and data mining methods to tackle them. With extensive experiments on well-known benchmark datasets and real-world university webpage datasets, we demonstrate that our search engine is better than traditional keywords-based search engine. In addition, two usability studies of SEEU in our university show that SEEU has a great promise for university search.

## 8.2 Future Work

In our future work, we plan to improve our search engine in three directions.

- Firstly, we will incorporate more hierarchies to better capture users' search intention, such as campus life (i.e., clubs, recreation, news), people directory, regions of places in Canada, into SEEU. Some existing speciality search, such as video and image search, can also be introduced. In this way users can search across multiple topic hierarchies and different media types.
- Secondly, we plan to use active learning on crowdsourcing platforms (such as Amazon Mechanic Turk[1]) to improve hierarchical classification. We can use active learning to reduce the number of noisy training webpages crawled from search engines. In addition, active learning can also be used to correct webpage classification errors and thus improve hierarchical browsing.
- Thirdly, we will also study the problem of classification in dynamic (not fixed) topic hierarchies. When a new topic (e.g., hot event, new research area) is emerging, it is very likely that many webpages belonging to that topic can not be predicted into any subcategories but stay at the top categories. Detecting such emerging topics is crucial for maintaining the effectiveness of hierarchical classification.

Finally, we have reported our search engine to the Information and Technology Service[2] in our university. They are very interested in SEEU. We are now working with them to deploy SEEU in our university.

[1]`https://www.mturk.com/mturk/`

[2]Information and Technology Service is responsible for the network management and search engine development in Western University. Their home page is `http://www.uwo.ca/its`.

# Bibliography

[1] Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 280–290, New York, NY, USA, 2003. ACM.

[2] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Workshop on Machine Learning in the New Information Age*, pages 9–17, June 2000.

[3] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] Jayme Garcia Arnal Barbedo and Amauri Lopes. Automatic genre classification of musical signals. *EURASIP J. Appl. Signal Process.*, 2007(1):157–157, January 2007.

[5] Steven M. Beitzel, Eric C. Jensen, David D. Lewis, Abdur Chowdhury, and Ophir Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Trans. Inf. Syst.*, 25(2), April 2007.

[6] Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 33–44, New York, NY, USA, 2008. ACM.

[7] Paul N. Bennett and Nam Nguyen. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 11–18, New York, NY, USA, 2009. ACM.

[8] Paul N. Bennett, Krysta Svore, and Susan T. Dumais. Classification-enhanced ranking. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 111–120, New York, NY, USA, 2010. ACM.

[9] Wei Bi and James Kwok. Mandatory leaf node prediction in hierarchical multilabel classification. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 153–161. 2012.

[10] David Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[12] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML ’98, pages 55–63, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[13] David J. Brenes, Daniel Gayo-Avello, and Kilian Pérez-González. Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 workshop on Web Search Click Data*, WSCD ’09, pages 1–7, New York, NY, USA, 2009. ACM.

[14] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, 1998. Elsevier Science Publishers B. V.

[15] Klaus Brinker. On active learning in multi-label classification. In *From Data and Information Analysis to Knowledge Engineering*, pages 206–213. 2006.

[16] Andrei Broder. A taxonomy of web search. In *ACM SIGIR Forum*, volume 36, pages 3–10, New York, NY, USA, September 2002. ACM.

[17] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML ’05, pages 89–96, New York, NY, USA, 2005. ACM.

[18] Christopher J. C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, Microsoft Research, 2010.

[19] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML ’07, pages 129–136, New York, NY, USA, 2007. ACM.

[20] A. Cardoso-Cachopo and A. Oliveira. An empirical comparison of text categorization methods. In *String Processing and Information Retrieval*, pages 183–196. Springer, 2003.

[21] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17, 2009.

[22] Michelangelo Ceci and Donato Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *J. Intell. Inf. Syst.*, 28:37–78, 2007.

[23] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[24] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - Proceedings Track*, 14:1–24, 2011.

[25] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 145–152, New York, NY, USA, 2005. ACM.

[26] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[27] David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the 19th annual conference on Learning Theory*, COLT'06, pages 605–619, Berlin, Heidelberg, 2006. Springer-Verlag.

[28] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647. MIT Press, 2001.

[29] Steve Cronen-Townsend and W. Bruce Croft. Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*, HLT '02, pages 104–109, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[30] Wisam Dakka, Panagiotis G. Ipeirotis, and Kenneth R. Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 768–775, New York, NY, USA, 2005. ACM.

[31] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proceedings of the 6th international conference Recherche d' Information Assistee par Ordinateur*, RIAO '00, pages 302–313, 2000.

[32] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.

[33] C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *ISMIR '07*, pages 77–80, 2007.

[34] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[36] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM Comput. Surv.*, 27(3):326–327, September 1995.

[37] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 2000.

[38] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM, 2000.

[39] Andrea Esuli and Fabrizio Sebastiani. Active learning strategies for multi-label text classification. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 102–113, Berlin, Heidelberg, 2009. Springer-Verlag.

[40] C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. Automated categorization in the international patent classification. *SIGIR Forum*, 37:10–25, April 2003.

[41] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[42] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003.

[43] William B. Frakes and Ricardo Baeza-Yates, editors. *Information retrieval: data structures and algorithms*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[44] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003.

[45] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Trans. Inf. Syst.*, 7(3):183–204, July 1989.

[46] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, April 2004.

[47] Viet Ha-Thuc and Jean-Michel Renders. Large-scale hierarchical text classification without labelled data. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 685–694, New York, NY, USA, 2011. ACM.

[48] Eui-Hong Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD '01, pages 53–65, London, UK, UK, 2001. Springer-Verlag.

[49] B. Hayete and J.R. Bienkowska. Gotrees: predicting go associations from protein domain composition using decision trees. In *Pac Symp Biocomput*, volume 10, pages 127–138, 2005.

[50] M. Hearst. Design recommendations for hierarchical faceted search interfaces. In *ACM SIGIR Workshop on Faceted Search*, pages 1–5. Citeseer, 2006.

[51] Marti A. Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *Proceedings of the 2nd Workshop on Human-Computer Interaction and Information Retrieval*, 2008.

[52] Tomi Heimonen and Mika Käki. Mobile findex: supporting mobile web search with automatic result categories. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 397–404, New York, NY, USA, 2007. ACM.

[53] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[54] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI*, Stockholm, 1999.

[55] Thomas Hofmann and Jan Puzicha. Unsupervised learning from dyadic data. Technical Report TR-98-042, International Computer Science Insitute, Berkeley, CA, 1998.

[56] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM.

[57] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *Trans. Neur. Netw.*, 13(2):415–425, March 2002.

[58] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6:429–449, October 2002.

[59] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[60] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.

[61] Thorsten Joachims. Advances in kernel methods. chapter Making large-scale support vector machine learning practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[62] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[63] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[64] Amy K. Karlson, George G. Robertson, Daniel C. Robbins, Mary P. Czerwinski, and Greg R. Smith. Fathumb: a facet-based interface for mobile search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 711–720, New York, NY, USA, 2006. ACM.

[65] A. Kennedy and D. Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, 2006.

[66] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, August 1988.

[67] Da Kuang, Xiao Li, and Charles X. Ling. A new search engine integrating hierarchical browsing and keyword search. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2464–2469. AAAI Press, 2011.

[68] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 81–89, New York, NY, USA, 1998. ACM.

[69] Avraham Leff and James T. Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, EDOC '01, pages 118–, Washington, DC, USA, 2001. IEEE Computer Society.

[70] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 4–15, London, UK, UK, 1998. Springer-Verlag.

[71] D.D. Lewis, Y. Yang, T.G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[72] Chengkai Li, Ning Yan, Senjuti B. Roy, Lekhendro Lisham, and Gautam Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 651–660, New York, NY, USA, 2010. ACM.

[73] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS'07*, pages –1–1, 2007.

[74] Xiao Li, Da Kuang, and Charles X. Ling. Active learning for hierarchical text classification. In *Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'12, pages 14–25, Berlin, Heidelberg, 2012. Springer-Verlag.

[75] Xiao Li, Charles X. Ling, and Huaimin Wang. Hierarchical classification and its novel application in university search. *Submitted to IEEE Transactions on Knowledge and Data Engineering*, 2013.

[76] Xiao Li, CharlesX. Ling, and Huaimin Wang. Effective top-down active learning for hierarchical text classification. In *Proceedings of the 17th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, pages 233–244. Springer Berlin Heidelberg, 2013.

[77] Xiaoli Li, Rohit Joshi, Sreeram Ramachandaran, and Tze-Yun Leong. Classifying biomedical citations without labeled training examples. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pages 455–458, Washington, DC, USA, 2004. IEEE Computer Society.

[78] J.J. Lin and C. Dyer. *Data-Intensive Text Processing With MapReduce*. Synthesis lectures on human language technologies;. Morgan & Claypool, 2010.

[79] Wei-Hao Lin and Alexander Hauptmann. News video classification using svm-based multimodal classifiers and combination strategies. In *Proceedings of the tenth ACM international conference on Multimedia*, MULTIMEDIA '02, pages 323–326, New York, NY, USA, 2002. ACM.

[80] Nan Liu and C. Yang. Mining web site's topic hierarchy. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 980–981, New York, NY, USA, 2005. ACM.

[81] Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. In *ACM SIGKDD Explorations Newsletter - Natural language processing and text mining*, volume 7, pages 36–43, New York, NY, USA, June 2005. ACM.

[82] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and SašO Deroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recogn.*, 45(9):3084–3104, September 2012.

[83] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI Workshop on Learning for Text Categorization*, 1998.

[84] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[85] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1998.

[86] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[87] John C. Platt. Advances in kernel methods. chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[88] John C. Platt. Probabilistic outputs for support vector machines. *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[89] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[90] Michael Prince. Does active learning work? a review of the research. *Journal of Engineering Education*, 93(3):1–10, 2004.

[91] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41:12:1–12:31, February 2009.

[92] Daniel Ramage, Paul Heymann, Christopher D. Manning, and Hector Garcia-Molina. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM ’09, pages 54–63, New York, NY, USA, 2009. ACM.

[93] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.

[94] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. pages 109–126, 1996.

[95] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. volume 7, pages 1601–1626. JMLR.org, December 2006.

[96] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML ’01, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[97] Miguel E. Ruiz and Padmini Srinivasan. Hierarchical neural networks for text categorization (poster abstract). In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR ’99, pages 281–282, New York, NY, USA, 1999. ACM.

[98] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C.D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. *arXiv preprint cs/0106040*, 2001.

[99] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *J. ACM*, 15(1):8–36, January 1968.

[100] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[101] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 979–988, New York, NY, USA, 2010. ACM.

[102] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM.

[103] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In Thrun and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 937–944, Cambridge, MA, 2003. MIT Press.

[104] Ammon Shashua and Anat Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. *Neural Information Processing Systems*, 16, 2002.

[105] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, July 2006.

[106] Carlos N. Silla, Jr. and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72, January 2011.

[107] S. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, October 1997.

[108] Emilia Stoica and Marti A. Hearst. Automating creation of hierarchical faceted metadata structures. In *In Procs. of the Human Language Technology Conference (NAACL HLT)*, 2007.

[109] Aixin Sun. Blocking reduction strategies in hierarchical text classification. *IEEE Trans. on Knowl. and Data Eng.*, 16(10):1305–1308, October 2004.

[110] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 521–528, Washington, DC, USA, 2001. IEEE Computer Society.

[111] Vinhtuan Thai, Pierre-Yves Rouille, and Siegfried Handschuh. Visual abstraction and ordering in faceted browsing of text collections. *ACM Trans. Intell. Syst. Technol.*, 3(2):21:1–21:24, February 2012.

[112] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, 2002.

[113] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. In *Int J Data Warehousing and Mining*, volume 2007, pages 1–13, 2007.

[114] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, 73(2):185–214, November 2008.

[115] Karin Verspoor, Judith Cohn, Susan Mniszewski, and Cliff Joslyn. Categorization approach to automated ontological function annotation. In *Protein Science*, pages 1544–1549, 2006.

[116] Alexei Vinokourov and Mark Girolami. A probabilistic framework for the hierarchic organisation and classification of document collections. *J. Intell. Inf. Syst.*, 18(2-3):153–172, March 2002.

[117] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[118] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *Proceedings of the 25th European conference on IR research*, ECIR'03, pages 393–407, Berlin, Heidelberg, 2003. Springer-Verlag.

[119] Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 619–626, New York, NY, USA, 2008. ACM.

[120] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 917–926, New York, NY, USA, 2009. ACM.

[121] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, May 1999.

[122] Yiming Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 137–145, New York, NY, USA, 2001. ACM.

[123] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[124] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, July 2007.

[125] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 116–, New York, NY, USA, 2004. ACM.

[126] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6(3):3:1–3:24, 2010.

# Appendix A

# Nutch Crawler

We use the popular Apache *Nutch* crawler[1] in SEEU. *Nutch* is an open source web crawler written in Java. It has a highly modular architecture. Many useful plug-ins, such as data retrieval, hyperlink analysis and document parsing, are already integrated into *Nutch*. Thus, we can collect webpages in an automated manner, and reduce lots of implementation work.

We briefly describe the work flow of *nutch* crawlers. Given a crawling task (such as crawling webpages in Western), *Nutch* uses a Breadth-first search algorithm to crawl webpages. Algorithm 1 shows the pseudo code of *Nutch* crawling algorithm. It iteratively loops though four basic operations (*Generate*, *Fetch*, *Parse* and *Update*) until all the webpages are crawled.

```
Input: $S$: A list of initial URL seeds,
       $N$: the number of webpages to be fetched at each iteration,
       $T$: the maximal iteration,
       $R$: the URL matching rules
Output: All the parsed documents
1  Create the crawling database from $S$;
2  $t = 1$;
3  repeat
4      Generate the top $N$ most important webpages $W$ that satisfies $R$ from the crawling database;
5      if $W = \emptyset$ then
6          Quit;
7      end
8      Fetch the URLs in $W$ and estimate their importance scores;
9      Parse the crawled webpages in $W$;
10     Update the crawling database by injecting new URls;
11     $t \leftarrow t + 1$;
12 until $t > T$;
```

**Algorithm 1:** The pseudo code of *Nutch* crawling algorithm.

We list the details of the four operations as follows.

- *Generate*. The *Generate* command reads the crawling database, filters out unwanted URLs by the URL matching rules and picks up the top $N$ most important URLs in $W$

[1]The home page of *Nutch* is `http://nutch.apache.org/`.

(importance scores are computed in *Fetch* command). Specifically, when *Nutch* crawls webpages in a university, there usually exists many irrelevant webpages, such as webpages from Youtube and Facebook. In *Nutch*, we can set up URL matching rules to filter out these unnecessary webpages. For example, to restrict *Nutch* to crawl webpages in Western only , we can set up a Regex rule as "+ˆhttps?://([a-z0-9]*\.)*uwo.ca". In addition, as we mainly study text classification and retrieval in this thesis, we need to restrict *Nutch* to skip multimedia files (e.g., images, videos, compressed archives and so on). We also skip the URLs containing special characters (e.g., &,?,=) as dynamic queries. A sample of the URL matching rules that used for crawling webpages from Western can be seen in Table A.1.

Table A.1: A sample of the URL matching rules for crawling webpages from Western.

| Description | Regex Rule |
|---|---|
| Skip multimedia | -\.(gif\|jpg\|png\|ico\|css\|sit\|eps\|wmf\|zip\|rar\|bz2\|tar\|jar\|mpg\|gz\|rpm\|tgz\|mov \|exe\|jpeg\|bmp\|js\|sdc\|iso\|7z\|mp3\|flv\|avi\|AVI\|wmv\|pdb\|psd\|mp4\|wav\|sql \|deb\|vrdfits\|rdvfits\|fits\|debug\|class\|dmg\|dtd) |
| Skip queries | -[*!@&] |
| Only Western | +ˆhttps?://([a-z0-9]*\.)*uwo.ca |

- *Fetch*. The *Fetch* command crawls the webpages from the URL list $W$ and store the raw webpage content in a local database. When *Nutch* visits a webpage, the *Fetch* command updates its importance score by the OPIC (Adaptive On-line Page Importance Computation) algorithm [1].

  The basic idea of OPIC algorithm is to model the web as a simple finical system. The importance score of each webpage is modeled as the total number of *virtual cash* flowing through it during crawling. Specifically, OPIC initializes a cash account for each webpage. When visiting a webpage, OPIC add its current cash to its history, distribute the cash uniformly to all its adjacent webpages and clear its account. After a finite number of iterations, the importance score of each webpage is computed as the sum of history cash and the current cash.

  The importance score is used for *Generate* command to bias the selection of the top $N$ most important webpages. It can also be used as an approximation of the PageRank [14] scores as proved in [1].

- *Parse*. The *Parse* command parses the crawled webpages and extracts text data. *Nutch* has already integrated many parser plug-ins to process different file types, such as PDF, DOC and XML. When parsing a file type, *Nutch* will call the corresponding parser plug-in to extract the text data.

- *Update*. The *Update* command updates the crawling database by injecting new URLs extracted by the *Fetch* command.

# Curriculum Vitae

| | |
|---|---|
| **Name:** | Xiao Li |
| | |
| **Post-Secondary** | University of Western Ontario |
| **Education and** | London, ON |
| **Degrees:** | 2009 - 2013 Ph.D. |
| | |
| | National University of Defense Technology |
| | Changsha, Hunan, China |
| | 2006 - 2008 M.Sc. |
| | |
| | National University of Defense Technology |
| | Changsha, Hunan, China |
| | 2002 - 2006 B.Sc. |
| | |
| **Awards:** | First prize award in Data Mining Session of |
| | UWO Research in Computer Science, UWORCS 2013 |
| | Faculty of Science |
| | University of Western Ontario |
| | London, ON |
| | April, 2013 |
| | |
| | First prize award in Data Mining Session of |
| | UWO Research in Computer Science, UWORCS 2012 |
| | Faculty of Science |
| | University of Western Ontario |
| | London, ON |
| | April, 2012 |
| | |
| | First prize award in Software Demo Session of |
| | UWO Research in Computer Science, UWORCS 2012 |
| | Faculty of Science |
| | University of Western Ontario |
| | London, ON |
| | April, 2012 |

| | |
|---|---|
| **Scholarship:** | Western Graduate Research Scholarship<br>2009 - 2013 |
| | China Scholarship Council Scholarship<br>2009 - 2013 |
| **Related Work Experience:** | Research Assistant<br>University of Western Ontario<br>London, ON<br>2009 - 2013 |
| | Teaching Assistant<br>University of Western Ontario<br>London, ON<br>2009 - 2013 |

**Publications:**

1. **X. Li**, C.X. Ling and H. Wang. Hierarchical Classification and its Novel Application in University Search. IEEE Transactions on Knowledge and Data Engineering, 2013 (being reviewing).

2. **X. Li** and C.X. Ling. Effective Top-down Active learning for hierarchical text classification. In Proceedings of the 17th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part II, Springer-Verlag, 2013, 233-244

3. **X. Li**, D. Kuang, and C.X. Ling. Active learning for hierarchical text classification. In Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I, Springer-Verlag, 2012, 14-25

4. D. Kuang, **X. Li**, and C.X. Ling. A new search engine integrating hierarchical browsing and keyword search. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Three, AAAI Press, 2011, 2464-2469.