Structured Web Data Extraction: University Domain

by

Yifeng Li

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science
in

School of Computer Science
Carleton University
Ottawa, Ontario, Canada
October 2013

Abstract

In the Semantic Web [1], information is structured and thus processable by machines. However, it is still largely unrealized. The current web is simply a collection of unstructured documents. To find information on the web, we use search engines such as Google to retrieve relevant documents. Users often need to search through the retrieved documents to find information. Due to web information explosion, it has become harder and harder for users to find information easily. While Google is trying to provide the most relevant results, our goal is to provide precise results that answer structured queries. To achieve our goal, we adopt the information extraction approach. In particular, we extract structured data from the unstructured web and organize the extracted data in a database to provide search functions. This thesis focuses on the implementation of a web information extraction system in a university domain.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Mengchi Liu, who has taught me how to do real research and given me the opportunity to grow. I want to thank my parents for supporting and encouraging me in a way that no other ever could.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The World Wide Web is a network of interlinked hypertext documents accessed via the Internet and these interlinked hypertext documents are called web pages. Web pages are mainly intended for human viewers. Since data embedded in web pages is not organized in a pre-defined way and thus hard for programs to understand, web data is often referred to as unstructured data. Due to the information explosion on the web, search engines such as Google, Yahoo, and Bing have been developed to help us find information quickly. Hard as these search engines try to provide the most relevant results, users often need to search through lots of retrieved documents to find the desired information. This is due to the fact that the approach adopted by search engines leads to only relevant rather than desired exact results. As the web keeps expanding, it is increasingly tedious and time-consuming for users to find information on the web. In the envisioned Semantic Web, however, it consists of a "web of data" which is processable by machines. The Semantic Web community aims at converting the current web of unstructured documents into a "web of data", but this grand vision remains largely unrealized. On the good side, with the development of database technologies we are able to organize data in a database and search for desired information using structured queries. Data stored in a database is directly processable by machines and often referred to as structured data. Since

we can find information easily in a database, it is highly desirable for us to take advantage of database technologies to manage web data. However, we cannot directly apply database technologies on unstructured web data unless we can transform the unstructured web data into structured data. The transformation can be realized using web information extraction techniques. In the remaining sections of this chapter, we first explain what structured data is, then show major advantages of structured query over keyword search, next describe our specific problem and goal in details and finally conclude with the outline of this thesis.

1.1 Structured Data

The most common example of structured data is probably data records stored in a database. In a relational database, data is organized based on columns and rows. Such structured data is processable by machines. It is also easily accessible by human viewers. Examples of unstructured data include emails, documents, web pages, blogs and various media files. Such data cannot be directly captured using columns and rows.

1.2 Structured Query vs Keyword Search

Generally speaking, to query from a database is a structured query while to search from traditional search engines (e.g., Google, Yahoo, Bing) uses a keyword search. Other examples of keyword search include desktop search and email search. We compare structured query and keyword search using four aspects [2].

Result quality In structured query we use expressive queries to retrieve precisely what we want. In keyword search information retrieval is keyword-based and

semantics of the query is ignored. As a result, we are only able to obtain relevant results and need to further identify what we want from the retrieved results.

Result structure In structured query the result is structured and processable by machines. In keyword search the result is usually unstructured and hard for machines to understand.

Result order In structured query the order is determined by the order in which data records were inserted. In keyword search the result order is determined by the service provider and we have no control over it.

Result quantity In structured query only exact results are returned. In keyword search most results are irrelevant and thus ignored by users. Generating these irrelevant results incurs a lot of extra computational costs, which is not energy efficient.

1.3 Problem Description

Imagine we want to find the list of universities that offer computer science programs in the Greater Chicago Area. What can we do to figure that out? We may probably want to try Google and end up with non-satisfactory results. What if we want to get the list of data mining professors in the US and view their profiles one by one? The web does provide all this information, but the information is unstructured and scattered all over the web, which makes it impossible to do a structured query. If we have all information about universities, faculties, colleges, divisions, schools, institutes, departments and faculty members organized in a database, we will be able to retrieve exactly what we want by constructing a simple query statement. To solve this problem, we need to find a mechanism to map information embedded in web pages to data records stored in a database.

1.3.1 Goal

Our work is to extract structured data from unstructured web data and store them into database so that they can be queried. While many other domains including government, entertainment and sports face the same problem, we focus on the university domain (i.e., information from university websites) in this thesis. First of all, the university domain is what we are familiar with. Second, after we come up with a solution for the university domain, we can potentially extend it to other domains. Since information on the web is unstructured, it is not readily processable by machines. Besides, it is impractical for us to access the content management system that drives the generation of web pages for every university. Thus, we aim to obtain information in its structured form using information extraction techniques and organize it into a database based on a schema generalized from domain investigation so that desired information can be retrieved accurately.

1.3.2 Scope

We focus on extracting English websites in the university domain. In this thesis, we deal with both Canadian and US universities. The main source of data is the official website of each university. In particular, we try to extract the following information:

University General Information Introductory information about the university.

Academic Unit Information Information about various academic units within the university.

Faculty Information Basic information of faculty members in a school or department.

1.3.3 Scalability

When we make design decisions, we take the ability to scale to a large number of domain instances into account. For example, we aim to extract hundreds of English university websites in the future with minimal adjustment to our current system. In order to ensure good scalability, the entire extraction process is completely automated.

1.3.4 Data Accuracy

In order to ensure high utility of our system, we intend the system to be precision-oriented. Whenever we need to make a tradeoff between recall and precision of the extracted data, we always consider data precision first. Since the system is full automatic, we can keep the data up to date by running the system periodically (e.g., re-extract the data every month).

1.3.5 Problem Formulation

We want to convert a collection of interlinked web documents in university domain into a database of structured data. These web documents are universally accessible via university homepages. Given university website, we try to extract its faculties, colleges, divisions, schools, institutes, departments and faculty members under each academic unit. Extraction details are summarized in Table 1:

Table 1: Extraction Details

Information Type	Description
University general information	We call them the attributes of the university, including president, founding time, address and motto.
Academic Unit Information	We include the names and homepages of all its faculties, colleges, divisions, schools, institutes, departments and the hierarchical relationships between these academic units.
Faculty Member Information	We call them the attributes of faculty members, including name, homepage, photo, position, phone, fax, email.

1.4 Outline of the Thesis

Chapter 2 introduces what web information extraction is and discusses possible solutions to the problem. Chapter 3 gives a detailed summary of our investigation results for the university domain, based on which we derive our extraction algorithms. Chapter 4 discusses related work in the university domain. Chapter 5 explains our algorithms in details and provides corresponding experimental results. Chapter 6 introduces the database we use and demonstrates extracted information through a website. Chapter 7 outlines our future tasks. Chapter 8 states our main contributions and concludes the thesis.

Chapter 2

Web Information Extraction

Information extraction starts out from the natural language processing community to address the named entity recognition problem. It has since developed into a topic spanning machine learning, database, web, and information retrieval. To define what information extraction is, we take the definition from Grishman [3]:

> The identification and extraction of instances of a particular class of events or relationships in a natural language text and their transformation into a structured representation (e.g.,a database).

They also make a distinction between information retrieval (IR) and information extraction (IE) that IR retrieves relevant documents from collections while IE retrieves relevant information from documents. Most recently, as information on the web grows exponentially, web information extraction has been placed in the spotlight. The difference between traditional information extraction and web information extraction is that the input of natural language text is replaced by web pages. The web is becoming the largest data source. Web data is embedded in natural language text, lists, two-dimensional tables and other forms. However, the current web is unstructured, which makes it difficult for the vast amount of useful web data to be queried, manipulated, or made use of by other applications directly. Web information extraction

fits in as an attempt to bridge the gap between the current web and the envisioned Semantic Web as required by most corporate information systems. There are many applications that benefit from the web information extraction advances. Among the notable ones are comparative shopping and semantic annotation. There are many more applications discussed in [4].

2.1 Web Pages

Web pages encode information in HTML format and provide navigation to other web pages via hyperlinks. Web pages also include style sheets, scripts and media type of files to render the final view. To view a web page, we need to make use of a web browser such as Internet Explorer or FireFox. In contrast to formal natural language text documents, web pages often contain a lot of information other than the main content, such as navigation links, news, and advertisements. While such kind of information is useful for the browsing user, it has a tendency to complicate extraction tasks by machines. The good side of web pages is that information is encoded in HTML and thus relationships between different pieces of information can be implied from their encoding HTML tags. For example, information grouped in an HTML table or list can be similar in content or type.

2.1.1 Document Object Model

The World Wide Web Consortium defines the Document Object Model (DOM) as follows [5]:

> The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further

processed and the results of that processing can be incorporated back into the present page.

Figure 1 gives an idea what the HTML DOM looks like. The HTML DOM defines the objects and properties of all HTML elements, and the methods to access them. In simple terms, every web page can be parsed as a DOM tree that has a similar structure as the one in the figure. Then we can use the provided methods to search, retrieve and modify information in the web page as represented by its corresponding DOM tree. There are many HTML parser libraries that transform a web page from plain text representation to its DOM representation so that we can easily access and manipulate various information on the web page. The Java HTML parser Jsoup [6] is a good example of such libraries. It provides a very convenient set of APIs for extracting and manipulating data from real-world HTML documents.

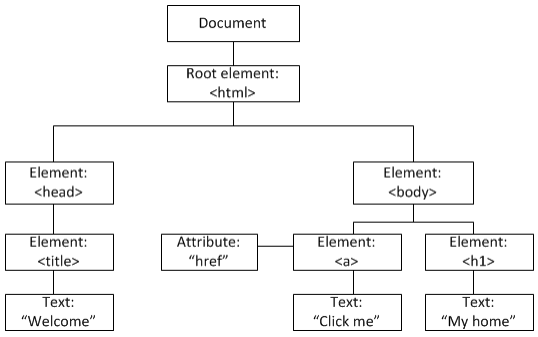


Figure 1: A minimum example that illustrates the Document Object Model.

2.1.2 CSS Box Model

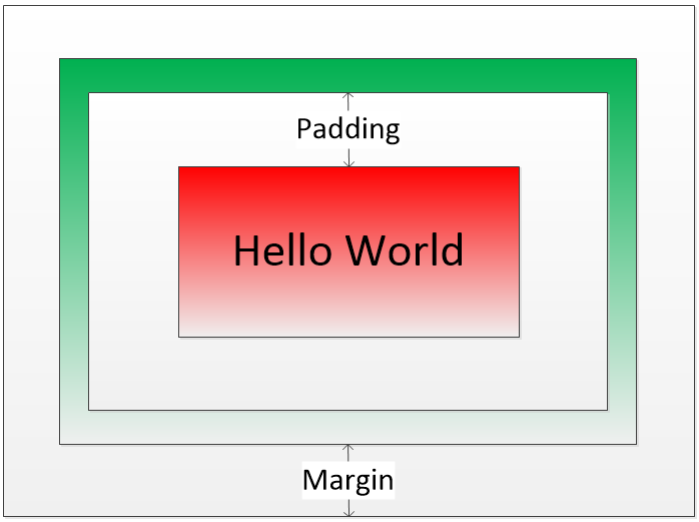


Figure 2: This image illustrates the CSS box model.

In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around HTML elements. It consists of four elements: margins, borders, padding, and the actual content [7]. All HTML elements can be considered as boxes. See Figure 2 for an illustration of the box model. The box in red represents the actual content while the part in green represents the border of the element. The box model determines the width and height of an HTML element when it is rendered by a layout engine, which is a major component of all web page rendering engines. The widths and heights of all HTML elements in a web page in turn determine the absolute and relative positions of these elements on the rendered page. Layout engines used by web browsers implement the box model and are able to retrieve information such as width, height, position of each HTML element on web pages. For example, CSSBox [8] is an HTML/CSS rendering engine written in pure Java that we can use to retrieve information about the rendered page's content, style and layout.

2.1.3 Web Crawler

A web crawler is a program that systematically browses the web. Web crawlers are usually used to index web documents by search engines. They are also used for extracting data from the web. A combination of policies are followed by a typical web crawler [9]:

Selection Policy decides which pages should be downloaded.

Re-visit Policy specifies when a page should be checked for changes.

Politeness Policy instructs how to avoid overloading a website.

Parallelization Policy defines rules for coordinating distributed web crawlers.

2.2 Recall and Precision

When we evaluate a work in web information extraction, there are quite a few dimensions to consider such as task domain, automation degree, techniques used, data completeness and accuracy. For data completeness and accuracy, the most important criteria are recall and precision [10], both of which come originally from the information retrieval community. High recall means that an algorithm returns most of the relevant results while high precision means that an algorithm returns substantially more relevant results than irrelevant. Recall and precision are computed as follows:

$$Recall = \frac{|\{relevant\,results\} \cap \{retrieved\,results\}|}{|\{relevant\,results\}|}$$

$$Precision = \frac{|\{relevant\,results\} \cap \{retrieved\,results\}|}{|\{retrieved\,results\}|}$$

There is a measure called F1-score that combines recall and precision. F1-score is computed as the harmonic mean of recall and precision, where recall and precision are evenly weighted:

$$F1 - score = \frac{2 \cdot recall \cdot precision}{recall + precsion}$$

2.3 Wrappers

A wrapper is a program that extracts content of a particular information source and converts it to structured data (i.e., in its relational form) [11]. Many web pages contain structured data, which are usually attributes of objects retrieved from underlying databases. These object attributes are typically displayed in web pages following some fixed templates. A wrapper is commonly used to extract structured data from such template-based web pages. Formally, a wrapper is a function from a page to the set of tuples it contains. There are two main approaches to wrapper generation: wrapper induction and automated data extraction.

2.3.1 Wrapper Induction

This approach is also called supervised information extraction in that manually labeled examples are needed for extraction rule learning. Since an expert is often needed to label or annotate a lot of web pages (training examples), there are some

major disadvantages with wrapper induction. First of all, the labeling process is tedious, time-consuming and error prone. In addition, it does not scale well. When we want to extract data from numerous sites, each site may have its own templates and thus require separate manual labeling. Second, it is very difficult to maintain these wrappers. Some sites such as product directory websites may change their page templates often. Whenever a site changes, wrappers generated for the site become obsolete. Wrapper maintenance is a major issue for this approach. Several major works will be discussed.

WIEN

WIEN [12] introduces a family of six wrapper classes, namely Left-Right (LR), Head-Left-Right-Tail (HLRT), Open-Close-Left-Right (OCLR), Head-Open-Close-Left-Right-Tail (HOCLRT), Nested-Left-Right (N-LR) and Nested-Head-Left-Right-Tail (N-HLRT) for web data extraction. The first four wrappers are used for semi-structured documents while the remaining two are used for hierarchically nested documents. The LR wrapper is a vector of 2K delimiters for a site containing K attributes. The HLRL class uses two additional delimiters to skip over potentially-confusing text in either the head or tail of the page. The OCLR class uses two additional delimiters to identify an entire tuple in the document. The HOCLRT wrapper combines OCLR and HLRT. N-LR and N-HLRT are extensions of LR and HLRT for extracting nested data.

STALKER

STALKER [13] is an IE system that focuses on hierarchical data extraction. To describe the structure of web documents, it introduces the concept of embedded catalog (EC). In EC description a page is a tree-like structure where internal nodes are lists of tuples and leaves are attributes to be extracted. There are two kinds of

rules to be learned in STALKER: 1) for each list node, a list iteration rule is required to decompose the list into individual tuples. 2) for each node in the tree, a rule to extract this node from its parent is required. The extraction rules are generated by using a sequential covering algorithm to cover as many positive examples as possible.

Summary

Other similar works include SoftMealy [14], WHISK [15], SRV [16], ViDE [17]. Wrapper induction is considered as a supervised IE approach. They take a set of web pages labeled with examples of the data to be extracted and output a wrapper. The user provides an initial set of labeled examples and the system may suggest additional pages for the user to label. For such systems, general users instead of programmers can be trained to use the labeling tool, thus reducing the cost of wrapper generation.

2.3.2 Automated Data Extraction

This approach is often considered unsupervised information extraction. Automated data extraction is possible because most web objects follow fixed templates. There are two categories of template-based pages.

Multiple Data Records Within One Page There are two or more data records within one single page.

Only One Data Record Per Page One page fits in only one data record.

Examples of the first category can be faculty list pages and product list pages. The information for each faculty member and each product item in the list is one data record from a database. Therefore, there are multiple data records in a single page and their templates or repeating patterns can be deduced using only one page. Examples of the second category can be a faculty member's introduction page and a product

item's detailed description page. Each of those pages contains only one data record from a database. In order to deduce their templates, we need at least two such pages to work on. By using unsupervised pattern mining techniques, we are able to discover both categories of templates. This in turn enables us to perform extraction automatically. Several major works will be discussed.

MDR

MDR [18] is proposed to extract data records from web pages. It assumes that there are two or more data records in a data region of a single web page. In other words, it only works on template-based pages. It is based on the belief that each data record in a data region is encoded with the same or similar sequence of tags. The algorithm works in three steps. First, it builds an HTML tag tree of the web page. Second, they compare tag strings of child nodes under the same parent using a string edit distance measure. If the similarity is greater than a predefined threshold, the nodes are counted towards a data region. Finally, data items are segmented from each data record. In a later work called DEPTA [19], they extend MDR by replacing the string edit distance measure with a tree edit distance measure and proposing a partial tree alignment technique to segment data items. The tree edit distance is more effective in measuring the similarity of two nodes in that it is able to preserve the tree structure of the nodes. However, a major drawback with DEPTA is that it fails to handle nested data records. A new enhancement is introduced in NET [20] to overcome this drawback by incorporating visual cues, performing a post-order traversal of the visual-based tag tree, and matching subtrees using tree edit distance.

RoadRunner

RoadRunner [21] only works on template-based pages as well. While MDR works on the first category of template-based pages, RoadRunner is capable of dealing with

the second (the two categories of template-based pages are summarized shortly). In RoadRunner, each generated web page is regarded as strings of HTML code. To extract data from a web page is equivalent to inferring a grammar for the HTML code. By comparing HTML pages of the same class, they generate a wrapper based on their similarities and differences. In particular, they use the ACME technique to align matched tokens and collapse for mismatched tokens. String mismatches are used to discover attributes while tag mismatches are used to discover iterators and optionals.

Summary

Other similar works include IEPAD [22], EXALG [23], DeLa [24]. This kind of IE system work on template-based web pages. Such pages are usually generated with a set of fixed templates filled by data retrieved from databases. As a result, the structured data from databases are embedded in the unstructured web pages. Since it is normally impossible for the public to extract data directly from databases, we need methods to recover structured data from web pages. This is how unsupervised IE systems come into play.

2.4 Adaptive Information Extraction

Wrappers are good at handling highly structured web pages such as product catalogues and telephone directories. However, they are usually incapable of dealing with less structured text. Such kind of web pages are very common on the web. For example, the customized homepage of a faculty member is considered to fall into the category of less structured web pages. The information of the faculty member is embedded in various kinds of forms such as tables, lists, and natural language paragraphs. In other words, the content of the page can vary from well-structured to

almost free text. Common wrappers often fail in the case of mixed text types. To address this problem, we need shallow natural language processing to be applied to less structured text. There are two main approaches to extracting information from less structured or even free text web pages: a classification-based approach and a sequential labeling-based approach.

2.4.1 Classification-Based IE

In this approach, information extraction is regarded as a classification problem. A classification model usually consists of two stages: learning and prediction. In the learning stage, we try to find a model for the labeled data that distinguishes it from other data. We later use the learned model in the prediction stage to identify an unlabeled instance as true or false. In a classification-based approach, we need to train a model per label (i.e., for every type of data we want to extract), which becomes tedious and hard to maintain when the number of labels grows. Support Vector Machine (SVM) [25] is among the most popular methods for classification. Other models include Maximum Entropy [26], Adaboost [27], and Voted Perceptron [28].

2.4.2 Sequential Labeling-Based IE

In this approach, information extraction is regarded as a sequential labeling problem. A web page document is viewed as a sequence of tokens. To each token we assign a sequence of labels to indicate the property of the token. It is similar that in natural language processing tasks we assign labels to each word as its part-of-speech. Formally, given an observation sequence $x = (x_1, x_2, ..., x_n)$, the information extraction task as sequential labeling is to find a label sequence $y^* = (y_1, y_2, ..., y_n)$ that maximizes the conditional probability $p(y|x)$. The sequential labeling-based approach is able to model the dependencies between target information while the classification

based approach is only able to model each target independently. Among the widely used models are Hidden Markov Model [29], Maximum Entropy Markov Model [30], and Conditional Random Fields [31].

2.5 Rule-based IE vs. Corpus-based IE

There are generally two categories of approaches for web information extraction: rule-based approach and corpus-based approach [32]. The key difference between the two is how they build up domain knowledge and generate extraction rules. The rule-based approach is also referred to as automatic extraction where a domain expert is essential in building up the knowledge base and generating extraction rules. The corpus-based approach is also referred to as wrapper induction where we need to train a domain model with a lot of annotated examples and use the trained model to generate extraction rules. Table 2 is a detailed comparison between the two approaches, which serves as a guideline regarding which approach to take for real-world tasks.

Table 2: Compare Two Information Extraction Approaches

	Rule-based Approach	**Corpus-based Approach**
Manual Annotating	Very little	Need to annotate a lot of examples for training
Dealing with bad training examples	Domain expert can detect them	Might potentially affect the training model
Domain Description	Expert's knowledge	Annotations from the training examples
Training Model	Not required	Determining factor of the outcome, computationally expensive
Rule Learning	System designer induces the rules from observed examples	Need to choose a training model to learn from a lot of examples
Performance	Depends on input	Depends on input
Migrate to other domains	Need to start from scratch	Need to annotate a lot of examples from the new domain unless such corpus already exist

2.6 Leveraging Web Service APIs

A web service is a software function which is available over the web. Web services have an interface described by the Web Services Description Language (WSDL). Other applications communicate with the services via the Simple Object Access Protocol (SOAP) [37]. Using web service technologies, we are able to convert our applications into web application and publish its APIs on the web so that people can find them on the web and take advantage of them for their own applications. There are many web services available for the task of web data extraction. Some services are free of charge while others are provided at a cost. We discuss several services related to our extraction task.

2.6.1 Web Extractor

The Web Extractor [38] provides APIs that we can use to crawl, structure, and normalize web data. It is also capable of delivering the data in various ways. For example, we can either write the output into files or send it to a search engine. The strengths of their services include high data precision, good scaling performance, and ability to deal with data complexity. Among their advanced features are search form filling, Javascript simulation, and named entity recognition.

2.6.2 Diffbot

Diffbot [39] provides services that help us understand and extract from web pages. It combines a variety of techniques such as computer vision and machine learning. Using their automatic APIs, we are only able to extract content from a limited set of page types including articles, blog posts, and product pages. In addition, they provide a custom API toolkit, using which we can create custom rules to extract any pages.

2.6.3 AlchemyAPI

AlchemyAPI [40] focuses on web services that use natural language processing techniques. It provides as many as 11 functions including named entity extraction, keyword extraction and relation extraction. Besides supporting all major programming languages, it is able to provide responses in various formats such as XML, JSON and RDF. The key advantage of AlchemyAPI is that we can use simple natural language queries to mine web pages so knowlege of technologies such as HTML and DOM is not required. Last but not least, it offers a suite of structured data extraction APIs which are built based on both structural and visual features. The drawback is that these APIs come at a high cost and cannot be adjusted for our specific use.

Chapter 3

Domain Investigation

We investigate 26 Canadian university websites and 74 US university websites and generalize shared features by these websites. The 26 Canadian universities are all research or comprehensive universities that have an English website. The 74 US universities are all well-known top universities. In this chapter, we summarize our findings. In particular, we try to give a comprehensive idea of the following:

Academic Units What are the names of various academic units within a university? What are the hierarchical relationships between these academic units?

Location of Academic Units Where can we find these academic units from university websites? How is academic unit information presented in web pages?

Faculty Members Under what academic units can we find faculty member information? How many kinds of faculty members are there?

Location of Faculty Members Where can we find faculty member information from academic unit websites? How is faculty member information presented in web pages?

3.1 Academic Units and Their Relationships

A Canadian or US university can have the following academic units: faculties, colleges, divisions, schools, institutes and departments. Some universities may have campuses and non-academic colleges. Universities normally have administrative departments as well. Table 3 shows examples of each kind.

Table 3: Examples of Various Units

Academic Units	**Examples**
Faculties	Faculty of Arts and Science, Faculty of Medicine
Colleges	College of Arts and Science, College of Medicine
Divisions	Division of Humanities, Division of Physical Sciences
Schools	School of Computer Science, School of Business
Institutes	Institute of Computer Science, Institute of Physics
Departments	Department of Computer Science, Department of Biomedical Engineering
Campuses	Vancouver Campus, University of Toronto Mississauga
Non-academic Colleges	Innis College, Victoria College
Administrative Departments	Human Resource Department, Department of Security Services

Campuses, non-academic colleges and administrative departments are not considered in this thesis. Research-oriented centers are not considered either. We focus on academic units including faculties, colleges, divisions, schools, institutes and departments. Based on our investigation, we find that the term faculty as a kind of

academic unit is used exclusively in Canadian universities. In US universities, the term college is used instead as an equivalent of the term faculty. For the sake of convenience, we will use the term college hereafter consistently to refer to both college and faculty. In addition, the term school and institute seem to have duality as well and we will use the term school to refer to both of them. Figure 3 demonstrates all possible hierarchical relationships between various kinds of academic units within a university.

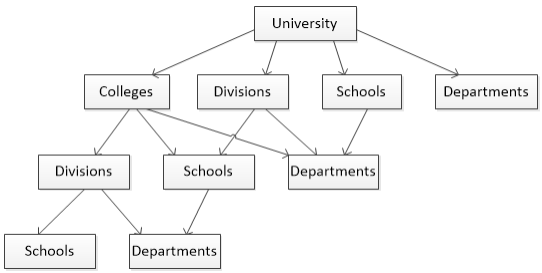


Figure 3: Hierarchical relationships between various kinds of academic units

In addition to what the figure shows, we also observe the following:

Colleges Colleges are always at the top level in the hierarchy. They represent the primary classification of academic units within a university.

Divisions If divisions are present within a university, they either function exactly like colleges as the primary classification of academic units or appear just under colleges in the hierarchy as a further classification of academic units. In the latter case, there are typically departments or schools under divisions as the final classification of academic units.

Schools Schools can function either as an upper-level classification or as a lower-level classification in the hierarchy. For example, the School of Law and School of Medicine are normally the result of a primary classification within a university

while the School of Computer Science and School of Biomedical Engineering are normally the result of a secondary classification under colleges.

Departments Departments are always at the bottom level in the hierarchy and function as the final classification of academic units within a university.

3.2 General Web List

Before going into contents and presentation styles of academic units on university websites, we first give a unified definition of a general web list. We adopt that of Gatterbauer et al. [41] as our definition of a list:

> A list is a series of similar data items or data records. A list can be either one-dimensional or two-dimensional; in both variants, we do not know the relationships between individual list items except for a possible ordering of the items.

By definition, any items that are either similar in type or similar in content can form a list. These items are usually parallel structures that can be easily recognized by human viewers. A list can not only be encoded in HTML list (i.e., with the ul or ol tag) but also in seemingly unrelated tags such as the p tag and the div tag. An HTML table can be viewed as a two-dimensional list as stated in the definition.

3.2.1 Classification Based on Visual Cues

We further classify general web lists into three categories based on visual cues:

Vertical List Each item is vertically aligned. In other words, all items in the list share the same x-coordinate.

Horizontal List Each item is horizontally aligned. In other words, all items in the list share the same y-coordinate.

Tiled List The list consists of multiple rows and columns to form a rectangle. Items on the same row and column are horizontally aligned and vertically aligned respectively.

Figure 4 gives an illustration on the three kinds of lists. Items in blue are grouped in a vertical list. Items in green are grouped in a horizontal list. Items in black are grouped in a tiled list.

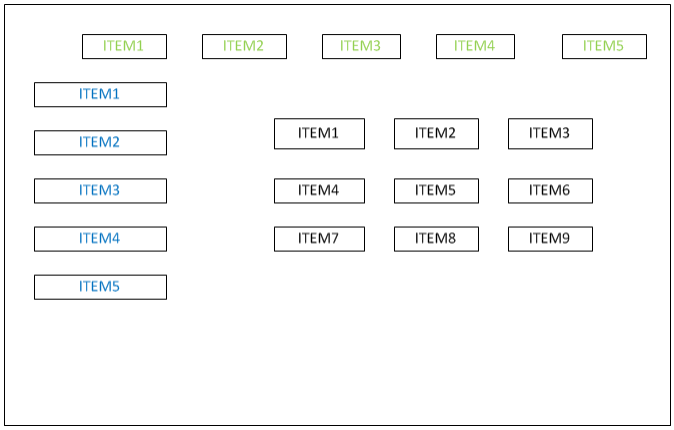


Figure 4: Three kinds of general web lists

3.2.2 Classification Based on Content Cues

We further classify general web lists into two categories based on content cues:

Simple List Each item contains only one piece of information.

Complex List Each item contains multiple pieces of information.

We will see examples of both simple list and complex list when we cover details about academic units and faculty members in later sections.

3.3 Division List and Unit List

All academic units can be found in some kind of lists within a university website. We assume so throughout the entire thesis as the foundation of our work. Depending on whether it is an upper-level classification or a lower-level classification, we classify all lists of academic units into two categories: division list and unit list.

Division List In a division list, we can have colleges, divisions and schools. We use the term division list to refer to a list of academic units which are the primary or upper-level classification of academic units within a university. For example, the College of Engineering and School of Business belong to a division list.

Unit List In a unit list, we can have schools and departments. We use the term unit list to refer to a list of academic units which are the secondary or lower-level classification of academic units within a university. For example, the School of Computer Engineering and Department of Biomedical Engineering belong to a unit list under the College of Engineering.

Referring back to the hierarchy in Figure 3, we can see that all leaf-level academic units belong to unit list while all internal-level academic units belong to division list. We will see more examples of both division list and unit list in the following sections.

3.4 Division Lists

In this section, we talk about division and division list in its general sense, which means a division list can contain colleges, divisions and schools. Every university website has at least one division list that contains the names of all its divisions if the university has divisions at all. We first study various cases of division lists and then classify them according to their appearances on web pages.

3.4.1 Case Study

Figure 5, 6, 7 show three kinds of division lists. In each figure the division list is marked in a red rectangle. In Figure 5, the division names in the list are vertically aligned. Another list of 3 items are right below the division list. Two observations are that the division list and the one below it are vertically aligned and there is a bigger gap between the two lists than between items in the same list. This kind of division lists can be seen as a simple list since each list item contains only one piece of information (i.e., the division name and its underlying hyperlink). In Figure 6, the division names in the list form a matrix of two rows and three columns. Alternatively, we can say they are both vertically and horizontally aligned. This kind of division lists can be seen as a complex list since each list item contains multiple pieces of information (i.e., besides the division name and its underlying hyperlink, other division information is present in the list). In Figure 7, the division names are vertically aligned. However, the gap between two names is much bigger than the first case. We can see that there is a sublist under each name and the sublist contains department names or program names. This kind of division lists can be seen as a complex list as well.

3.4.2 Division List Generalization

We classify division lists into three categories according to their visual appearances on web pages:

Nested Division List This kind of division list is a vertical and complex list. Besides the division name and the underlying hyperlink, other information such as its affiliated departments or its detailed information is placed under each division name as a sublist.

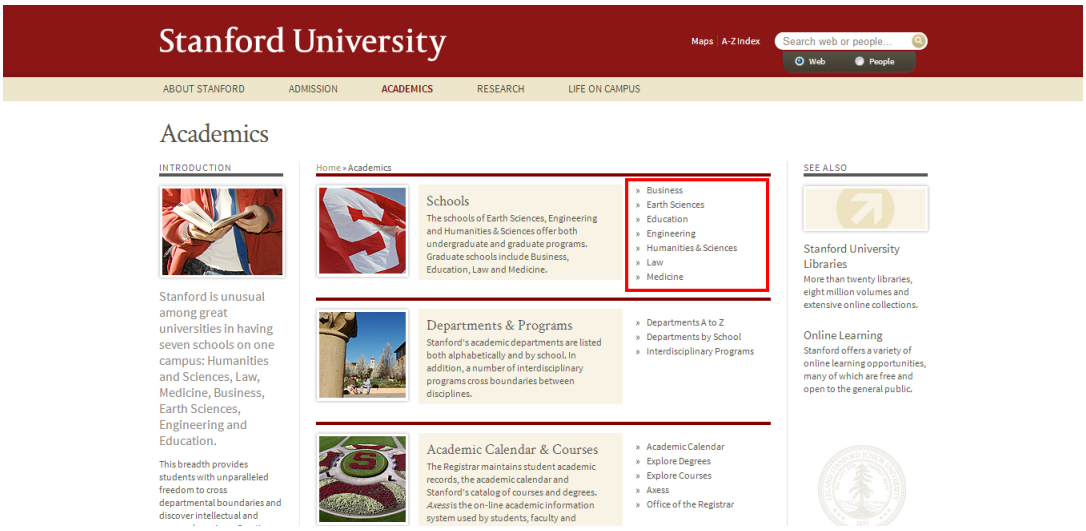


Figure 5: Stanford Division List

Indexed Division List This kind of division list is part of a bigger vertical list. There can be other lists that are vertically aligned with the division list. We need to further divide the bigger vertical list to retrieve the division list. There is typically a big visual gap between different lists that are vertically aligned.

Tiled Division List This kind of division list consists of multiple vertical lists to form the entire division list. The first item of each vertical list is horizontally aligned to each other.

The division list in Figure 7 is an example of nested division list. Under each division name, it has department information as its sublist. One important observation for a nested division list is that two division names in the list are far apart from each other vertically.

The division list in Figure 5 is an example of an indexed division list. We can see that there are other lists aligned vertically with the division list. To further illustrate the idea of an indexed division list, we make up the list layout on a web page in Figure 8. Before indexing, there are only three vertical lists on the page. After indexing, there are five vertical lists on the page and we are able to separate

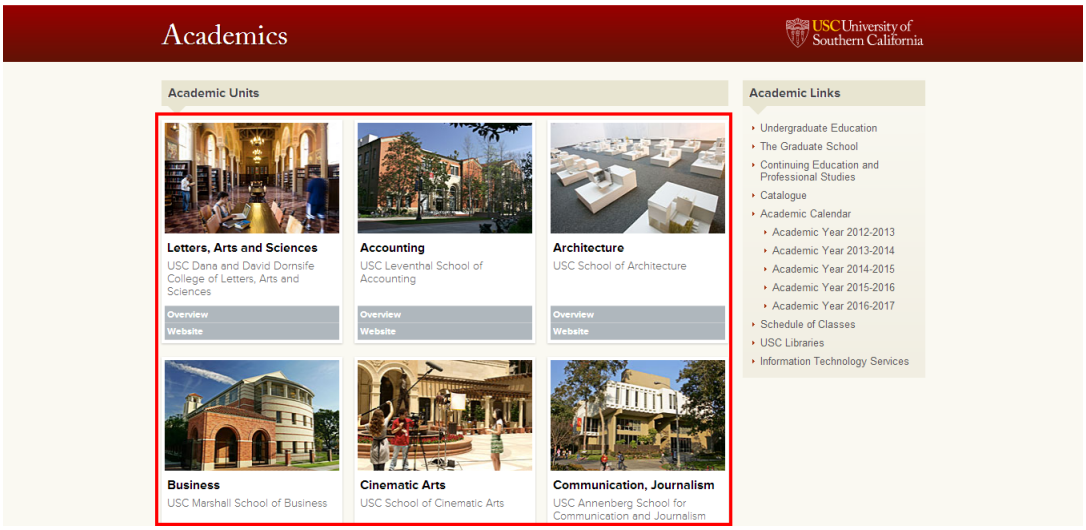


Figure 6: USC Division List

the division list from the bigger containing vertical list. The results of such indexing depend on the vertical distance between two different lists we use to index the lists. This is because items in the same list often have varying distances from each other. In other words, the distances between adjacent items are not necessarily the same. As a result, if the vertical distance between two different lists is not significant enough (e.g., it would be significant enough if it is more than two times bigger than the distance between adjacent items in the same list), the indexing does not work because the division list cannot be further separated from the bigger vertical list.

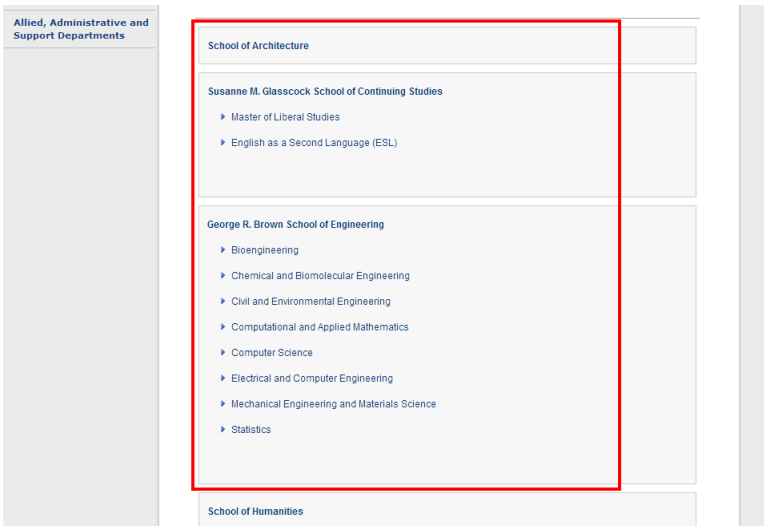


Figure 7: Rice Division List

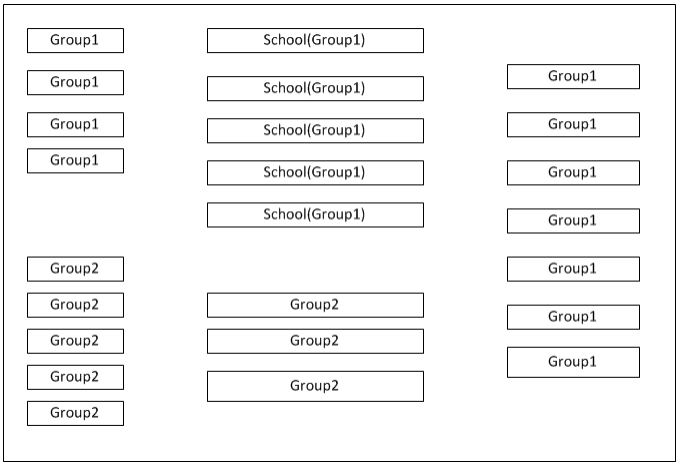


Figure 8: An illustration of indexed division list

The division list in Figure 6 is an example of tiled division list. The list has three columns and two rows where the division names in the same column are vertically aligned and the division names in the same row are horizontally aligned. The three requirements for this kind of lists are 1) there are at least two columns 2) the first column has at least two items 3) items on the first row are horizontally aligned. In particular, different columns do not necessarily contain the same number of items and items from the second row on do not need to be horizontally aligned.

3.4.3 Division Names

Table 4 lists division names of different formats.

Table 4: Various Kinds of Division Names

Case	Division Name	Description
1	School of Computer Science	This is the conventional way we write a school name
2	Tepper School of business	The school name is preceded by an extra modifier, whether it be a person or place name
3	Letters & Science, College of	The name is split by a comma with the name of the field first
4	Arts and Science	Only the name of the field is in the name

The set of keywords that can appear in a division name include “School”, “College”, “Faculty”, “Division”, “Institute”, “Center”, “Department”. The website designers usually use them in a consistent way, i.e., they are either present in all names of a division list or absent from all. There are a few exceptions where Case 4 is mixed with other three cases. Although such keywords as “Institute”, “Center”, “Department” may exist in one or two names of a division list, we consider them as negative words for identification of a division list. Chances are that if these keywords appear too frequently in a list, the list is probably not a division list. Rather, it may actually be a list of departments or a list of centers. We need to make a distinction between them.

3.5 How to Find Division Lists

In this section, we summarize how we can find the division list page from university homepage based on our investigation. There are several common keywords that lead us to division list page. These include but are not limited to "Academics", "Schools", "Colleges", "Campuses", "Divisions" and "Departments". Other keywords such as "Education" and "Programs" can serve the same purpose; however, they are much less common and do not always lead to division list pages in general. To be specific, we can locate the division list pages of most universities through at least one of the following paths:

Path One A division list is directly on the university homepage.

Path Two From the university homepage, follow a link containing either "Schools" or "Colleges". The landing page has a division list.

Path Three From the university homepage, follow a link containing "Academics", "Academic Units", "Academic Divisions" or other similar keywords. The landing page has a division list.

Path four If we do not find a division list on the landing page of Path Three, we start from the landing page and follow a link containing either "Schools" or "Colleges". The new landing page has a division list.

3.6 Unit Lists

In this section, we talk about unit and unit list in its general sense, which means a unit list can contain schools and departments. Unit lists are similar to division lists. We summarize the differences between the two.

3.6.1 Unit List Generalization

We classify unit lists into four categories instead of three categories. Besides nested list, indexed list, tiled list we define for division list, we add another category named horizonal list. Figure 9 gives an example of horizontal unit list. In a horizontal list, all list items have the same y-coordinate. In the case of a nested unit list, the sublist is usually a list of programs under the unit or detailed information about the unit.

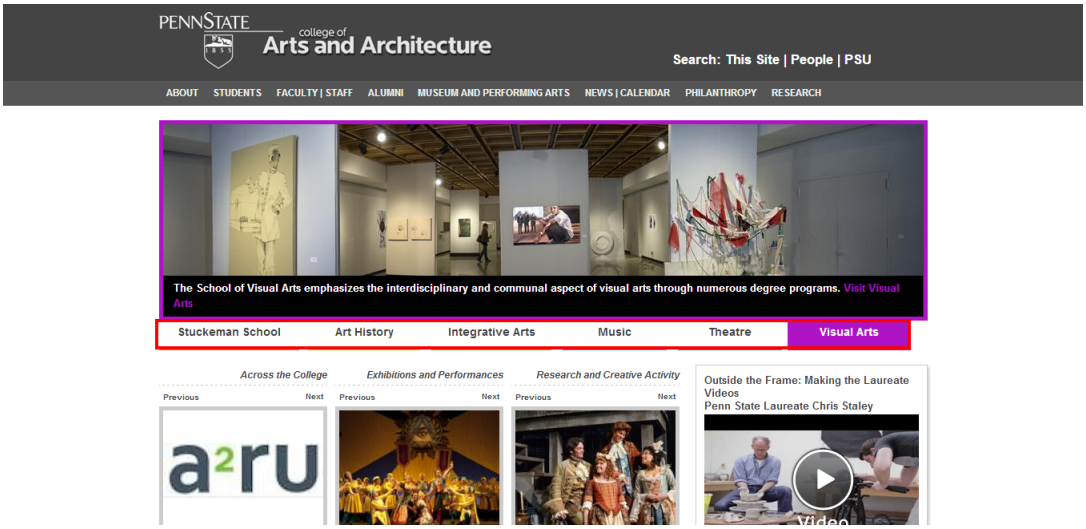


Figure 9: Horizontal Unit List Example

3.6.2 Unit Names

Table 5 lists unit names and possible higher-level divisions they belong to.

Table 5: Various Kinds of Unit Names

Case	Unit Name	Higher-level Division
1	School of Computer Science	Faculty of Science
2	Department of Finance	School of Business
3	Physics	College of Natural Sciences
4	Accounting, Department of	School of Business

The main keywords with a unit name is “School” and “Department”, which may or may not be present in a unit name. Other keywords including “Center”, “Institute”, and “Program” can appear in one or two unit names of the list. If they appear too often in the same list, the list is probably not a unit list. Unit list and program list are often placed in the same page. There are many terms such as “B.A.”, “M.S.”, “Ph.D.”, “MBA”, “Master” and “Doctorate”, which are typically seen in a program list. These terms can help distinguish a unit list from a program list or other undesirable lists (e.g., major list, minor list, degree list). We call them negative words of unit list. See Figure 10 and 11 for two examples where unit lists are mixed with other lists.

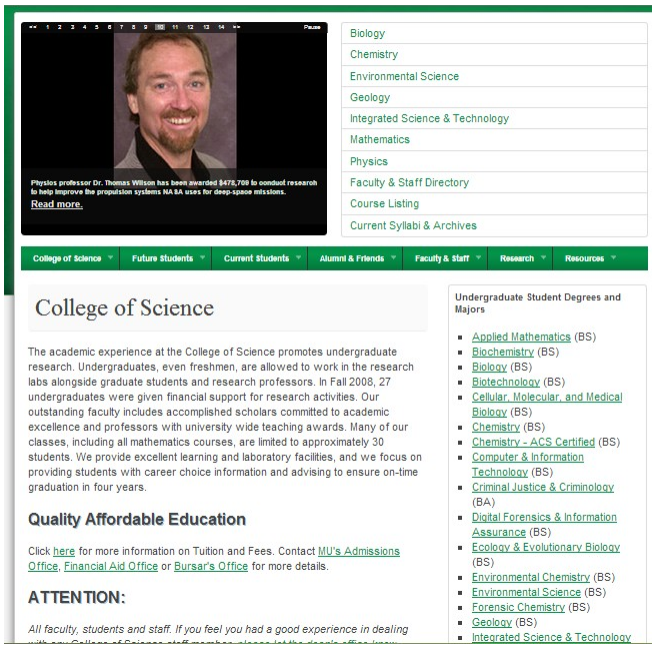


Figure 10: Unit list is mixed with major or degree list

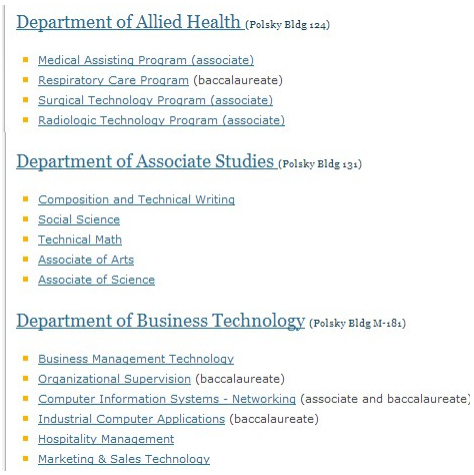


Figure 11: Unit list is mixed with program list

3.7 How to Find Unit Lists

Units typically belong to an upper-level division (it can be a college, a division, or a school). We investigate 150 divisions from different universities and summarize our findings on how to find units from a division homepage. The first important finding is that not all divisions are further divided into units. Two common examples are law schools and schools of medicine, which usually do not have smaller units. However, this is not a firm rule since we find out from our investigation that some law schools and schools of medicine do have units. Another important finding is that some divisions are further divided into smaller divisions first and then these divisions are in turn divided into units. One common example is College of Arts and Science, which is first divided into divisions such as Division of Arts, Division of Social Science, Division of Physical Science, and Division of Engineering. We focus on those divisions that directly have units and summarize how to locate unit list page within one particular division. The following is a summary of the most common paths we take from division homepages:

Path One A unit list is directly on the division homepage.

Path Two From the division homepage, follow a link containing either "Departments" or "Programs". The landing page has a unit list.

Path Three From the division homepage, follow a link containing "Academics", "Academic Units" or other similar keywords. The landing page has a unit list.

Path four If we do not find a unit list on the landing page of Path Three, we start from the landing page and follow a link containing either "Departments" or "Programs". The new landing page has a unit list.

Path five From the division homepage, follow a link containing "Schools", "Areas", "Fields". The landing page has a unit list.

The first three paths account for the majority of all scenarios. The keyword "Schools" appears a few times to indicate unit list page in cases where the upper-level division is actually a college.

3.8 Faculty Lists

We can find faculty directories under a unit homepage. If there are no units under a division, the division itself has a faculty directory. If there are units under a division, all these units have faculty directories and the division may or may not have a faculty directory directly under it. In case the division has it, it is a big directory that combines faculty members from all its units. We call faculty directories faculty lists throughout this thesis.

3.8.1 Case Study

Figure 12, 13, 14, 15 show four different cases of faculty list. In the first three cases, there are multiple pieces of information for each faculty member. We consider

these three kinds of lists as complex lists. In the last case, there is only one piece of information (i.e., faculty member name with an underlying hyperlink) for each faculty member. We consider this kind of lists as simple lists. In Figure 12, the information block for each faculty member is encoded by a sequence of various HTML tags like p, div and span and each faculty member has a similar sequence of HTML tags. In Figure 13, information for each faculty member is placed on a single row and the entire row is encoded by a tr tag. In Figure 14, there are 6 faculty members on a single row encoded by a tr tag. The 6 faculty members are in turn encoded by a td tag respectively within that tr tag. In Figure 15, the simple list is encoded by the ul tag and each faculty member is encoded by the li tag. Such kind of simple lists can be encoded by other tags as well.

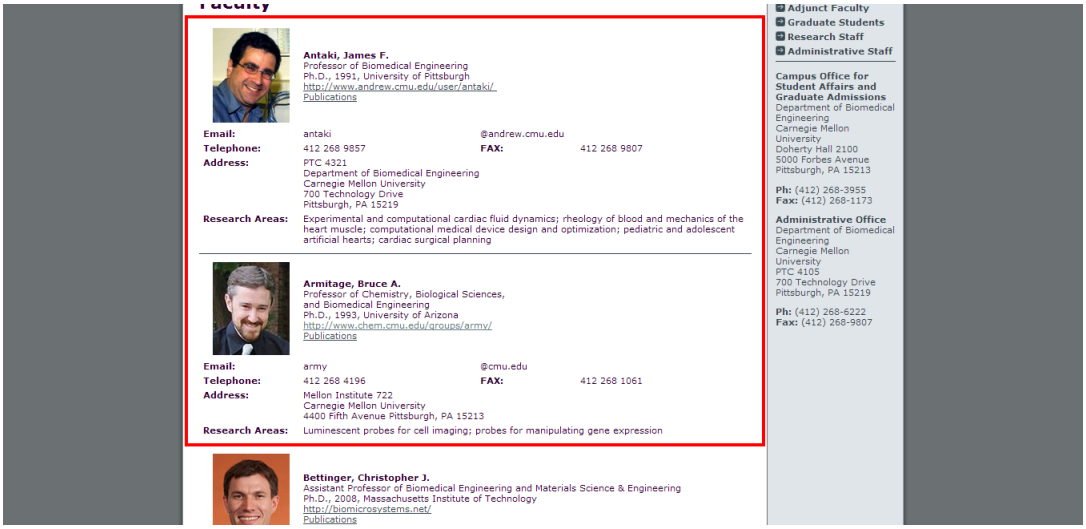
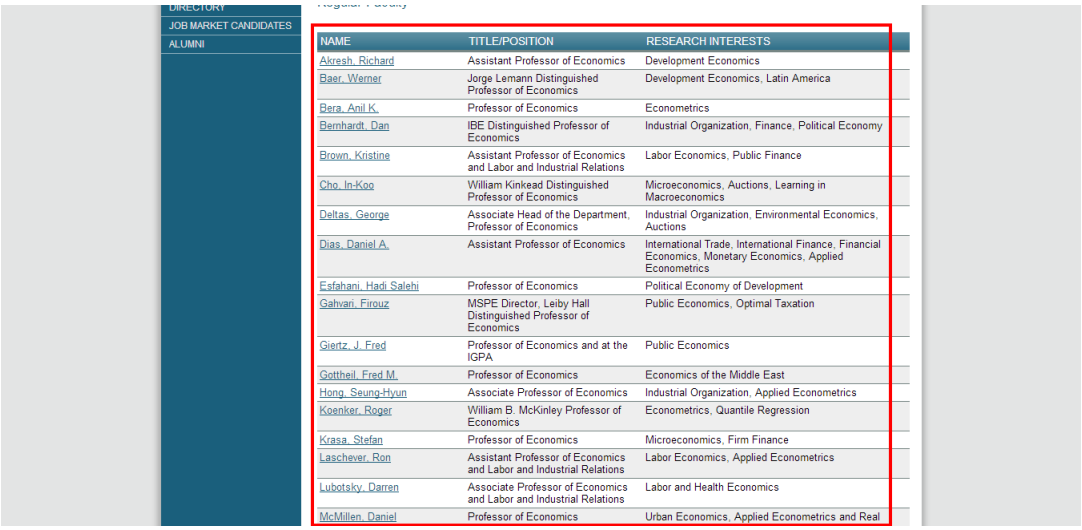


Figure 12: Faculty List Case One



DIRECTORY
JOB MARKET CANDIDATES
ALUMNI

NAME	TITLE/POSITION	RESEARCH INTERESTS
Akresh, Richard	Assistant Professor of Economics	Development Economics
Baer, Werner	Jorge Lemann Distinguished Professor of Economics	Development Economics, Latin America
Bera, Anil K.	Professor of Economics	Econometrics
Bernhardt, Dan	IBE Distinguished Professor of Economics	Industrial Organization, Finance, Political Economy
Brown, Kristine	Assistant Professor of Economics and Labor and Industrial Relations	Labor Economics, Public Finance
Cho, In-Koo	William Kinkead Distinguished Professor of Economics	Microeconomics, Auctions, Learning in Macroeconomics
Deltas, George	Associate Head of the Department, Professor of Economics	Industrial Organization, Environmental Economics, Auctions
Dias, Daniel A.	Assistant Professor of Economics	International Trade, International Finance, Financial Economics, Monetary Economics, Applied Econometrics
Esfahani, Hadi Salehi	Professor of Economics	Political Economy of Development
Gahvari, Firouz	MSPE Director, Leiby Hall Distinguished Professor of Economics	Public Economics, Optimal Taxation
Giertz, J. Fred	Professor of Economics and at the IGPA	Public Economics
Gottheil, Fred M.	Professor of Economics	Economics of the Middle East
Hong, Seung-Hyun	Associate Professor of Economics	Industrial Organization, Applied Econometrics
Koenker, Roger	William B. McKinley Professor of Economics	Econometrics, Quantile Regression
Krasa, Stefan	Professor of Economics	Microeconomics, Firm Finance
Laschever, Ron	Assistant Professor of Economics and Labor and Industrial Relations	Labor Economics, Applied Econometrics
Lubotsky, Darren	Associate Professor of Economics and Labor and Industrial Relations	Labor and Health Economics
McMillen, Daniel	Professor of Economics	Urban Economics, Applied Econometrics and Real

Figure 13: Faculty List Case Two

Steven M. Block
Research Fields: Biophysics, Nano Sci/Eng, Precision Measurement

Philip H. Bucksbaum
Research Fields: AMO Physics, Laser Physics, Ultrafast Science, X-Ray Physics

Robert L. Byer
Research Fields: Accelerator Physics, AMO Physics, Astrophysics, Laser Physics, Materials Science, Nano Sci/Eng, Photonics, Precision Measurement, Quantum Metrology, Synchrotron Radiation, Ultrafast Science, X-Ray Physics

Michel Digonnet
Research Fields: Control and Estimation, Electrical Engineering, Information Sci/Tech, Laser Physics, Materials Science, Nano Sci/Eng, Photonics, Precision Measurement

Sebastian Doniach
Research Fields: Biophysics, Condensed Matter, Imaging, Nano Sci/Eng, Statistical Physics, Synchrotron Radiation

Ian R. Fisher
Research Fields: Condensed Matter

Daniel S. Fisher
Research Fields: Biophysics, Condensed Matter, Statistical Physics

Martin M. Fejer
Research Fields: Photonics

Surya Ganguli
Research Fields: Biophysics, Condensed Matter, Electrical Engineering, Information Sci/Tech, Statistical

Harold Y. Hwang
Research Fields: Condensed Matter, Electrical Engineering, Energy Sciences, Materials Science, Nano

Mark A. Kasevich
Research Fields: AMO Physics, Quantum Many-Body Physics, Quantum Metrology, Quantum Optics, Ultrafast

Aharon Kapitulnik
Research Fields: Condensed Matter, Precision Measurement

Figure 14: Faculty List Case Three

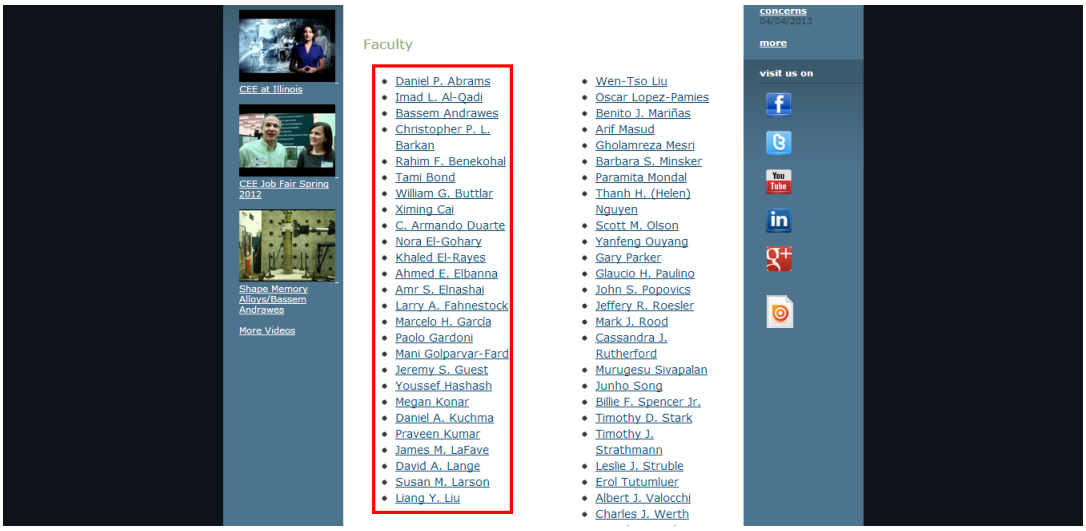


Figure 15: Faculty List Case Four

3.8.2 Faculty List Generalization

We first give a summary on the kinds of information that can appear in a faculty list. Then we classify faculty lists based on the way each faculty information block is encoded.

Faculty List Content

The one and only one piece of information shared by all faculty lists is faculty member's name. The majority of the faculty lists have a link for each faculty member that leads to their homepage. The amount of personal data presented in the list varies greatly from unit to unit. Some faculty lists are as simple as a name list with an underlying link for each name while some faculty lists contain as many as over 10 pieces of information for each faculty member. In the same list each faculty item consistently contains the same set of information. One exception is that some pieces of information can be absent from the list for some faculty members and present in the list for other faculty members. We give a summary for the most popular pieces in Table 6. We call each kind of information an attribute of faculty member.

Table 6: Faculty Attributes in Faculty List

Attribute	Description	Indicators
Name	The name of the faculty member	Bold, on the first line
Photo	The profile image of the faculty member	At top left or top right, with a face in it
Position	The position the faculty member holds	In the neighborhood of the name
Email	The departmental email address	Preceded by "Email", "Mail", "E-mail Address", etc.
Phone	The departmental telephone number	Preceded by "Phone", "Tel", "Voice", etc.
Fax	The departmental fax number	Preceded by "Fax", "Office Fax", "facsimile", etc.
Homepage	The personal page or introduction page URL	The hyperlink underlying the name, in the neighborhood of the name, or a link called "Homepage", etc.
Research	The research areas or fields	Preceded by "Research Areas", "Research Fields", "Research Interests", etc.
Office	The office location of the faculty member	Preceded by "Office", "Location", etc.
Address	The departmental mailing address	preceded by "Mailing Address", "Postal Address", "Mail to", etc.
Education	The education background information	Under the name, including Ph.D. degree and university

Faculty List Classification

We classify faculty lists into three categories according to the way each of their information blocks is encoded. The classification is based on the assumption that all information blocks are under the same parent node and in a contiguous information region. The first category is showed in Figure 16. We call it the one element per member pattern. In Figure 16, we can find four repeating units under the parent based on the tag sequences. Each unit consists of one single tag and each unit corresponds

to one faculty member. The faculty lists we have seen in Figure 13 and 15 belong to this category. It is the most commonly seen category for real-world faculty lists. The second category is showed in Figure 17. We call it the multiple elements per member pattern. In Figure 17, we can find two repeating units under the parent based on the tag sequences. Each unit consists of two tags and each unit corresponds to one faculty member. The faculty list we have seen in Figure 12 belongs to this category. The third category is showed in Figure 18. We call it the one element multiple members pattern. In Figure 18, we can find four repeating units under the parent based on the tag sequences. Each unit consists of one single tag and each unit corresponds to two faculty members. This means we can find nested repeating patterns inside each single tag. The faculty list we have seen in Figure 14 belongs to this category.

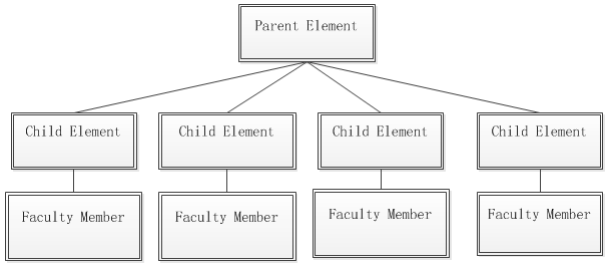


Figure 16: One Element Per Member

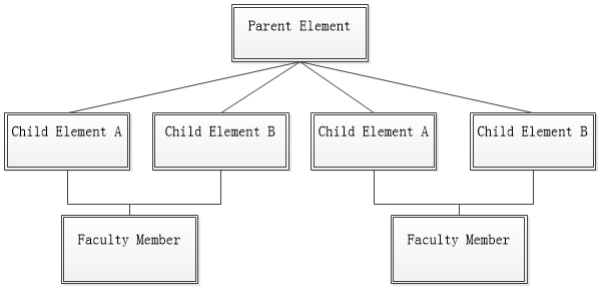


Figure 17: Multiple Elements Per Member

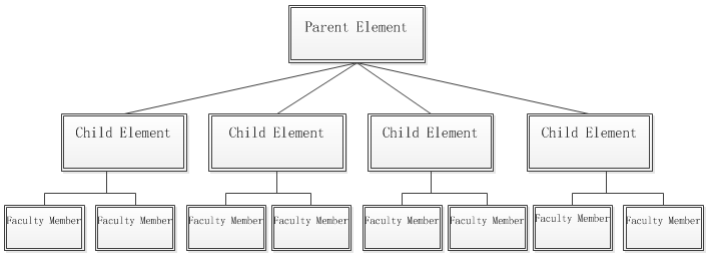


Figure 18: One Element Multiple Members

3.9 How to Find Faculty Lists

From a unit homepage, we are likely to find the faculty list by following links containing certain keywords such as “Faculty”, “People”, “Directory” and “Profiles”. The following are the most common links on the homepage:

1. Faculty & Staff Directory
2. Faculty and Staff Directory
3. Faculty/Staff Directory
4. Faculty & Staff
5. People
6. Faculty
7. Faculty listing
8. Faculty Profiles
9. Faculty Directory

We classify these links into two categories. The first 5 links fall into the first category. They often lead to pages that contain both faculty list and staff list or pages that

contain sublinks that lead to a faculty list page and a staff list page respectively. The remaining 4 links fall into the second category. They often lead to pages that contain a faculty list or pages that contain sublinks that lead to different types of faculty list pages. The most common sublinks include:

1. Core Faculty
2. Regular Faculty
3. Full-time Faculty
4. Voting Faculty
5. Affiliated Faculty
6. Visiting Faculty
7. Adjunct Faculty
8. Emeriti Faculty

In most cases the faculty list is one or two hyperlinks away from unit homepage. In cases where a unit has too many faculty members, the faculty list is divided into multiple smaller lists that are placed across multiple pages. These smaller lists are usually indexed alphabetically or numerically to form the entire faculty list.

3.10 Faculty Homepages

From each faculty list, we can normally find a link leading to each faculty member's homepage. By homepage, we mean a page that is dedicated to a particular faculty member and has basic information about the faculty member.

3.10.1 Case Study

Figure 20 and 19 show two kinds of homepages. In Figure 20, it is the faculty member's introduction page under the unit (i.e., department or school), where we can find the name, the profile photo, various kinds of contact information, position, education background and research interests of the member. In Figure 19, it is the faculty member's personal page which is probably designed by the faculty member himself. We can find almost the same set of information about the member as we find from his introduction page. Despite similar content on both pages, their presentation styles are quite different. In the introduction page, information is better formatted and the text used is also more formal. For example, "phone" may be used instead of "tel".

3.10.2 Faculty Homepage Generalization

There are two types of faculty homepage: template-based page and personalized page. Introduction pages fall into the first category while personal pages fall into the second. Template-based pages are usually generated from a backend faculty database. Introduction pages within the same unit share the same template and sometimes different units within the same division share the same template. Template-based pages only contain academics-related information including faculty name, profile photo, short bio, academic position, contact information, education background, research interest, publication, courses and students. However, in personalized pages, we can find other information such as family photos and non-academic hobbies. In template-based pages, attributes such as photo, fax and email are well formatted and the keywords used to indicate these attributes are more formal.

3.10.3 Comparison with Information in Faculty List

Some pieces of information are marked in rectangles in Figure 19. We have already seen some of the information in the faculty list, but there are some differences we need to summarize:

Data Set There can be shared pieces of information from both sources while each can have its own pieces of information.

Data Format The shared pieces of information from both sources can take different forms.

Data Location In the faculty list information for each member is usually found in a small information block while in faculty member's homepage information is scattered all over the page.

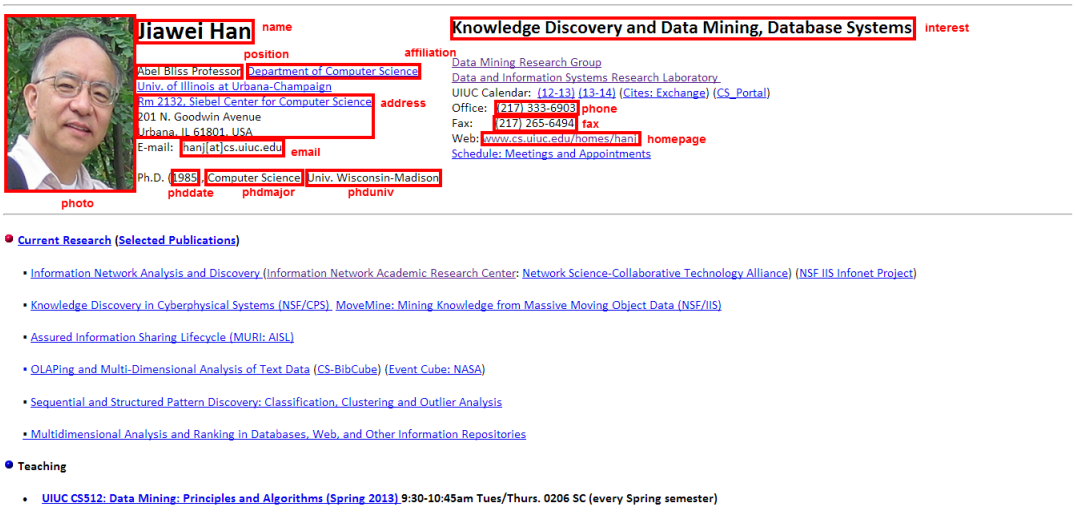


Figure 19: An example for faculty member's homepage

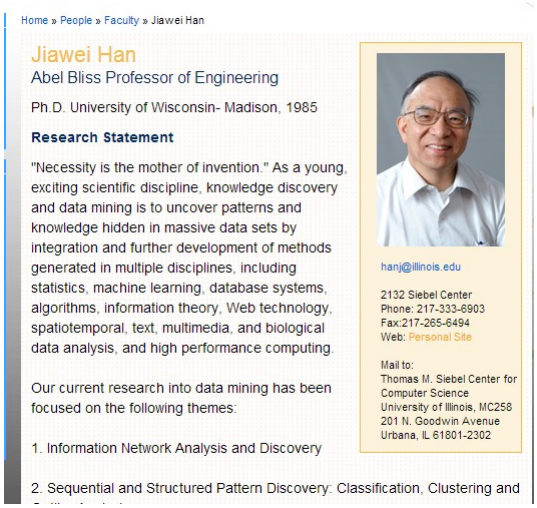


Figure 20: An example for faculty member's unit introduction page

Chapter 4

Related Work

In this chapter we discuss several works in the university or academic domain. We talk about two major ones in detail, from which we get inspirations and motivations for our own work.

4.1 Semantic Information Retrieval Using Ontology In University Domain

In [42], they try to achieve semantic information retrieval using ontology in university domain. By analyzing the query both syntactically and semantically by keyword expansion, they are able to re-rank and optimize the Google results for providing the relevant links. The search is made possible by construction of a strong ontology which forms the knowledge base. Their system eliminates the irrelevant results by forming refined queries and ranking the retrieved links. The drawback of this approach is that it still returns web documents (although more relevant) rather than the desired structured data, which can be directly consumed by users.

4.2 UniversityIE: Information Extraction From University Web Pages

In [32], they write an information extraction module and plug it into an existing web crawler. The information to be extracted is mainly general information about each university. For example, TOEFL requirement, president, address, deadlines, tuition are extracted. They adopt a ruled-based approach. To be specific, three kinds of rules are defined: positional, tabular, and syntax. Their approach is mainly based on natural language processing techniques. Integrated into the extraction module are two linguistic toolsWordNet and Link Grammar Parser. Page selection heuristics are proposed to make the crawler more efficient. They also make an effort to normalize domain phrases. For example, Test of English as Foreign Language is normalized to TOEFL.

4.3 An Information Extraction System For Heterogeneous Web Source

In [43], they build an information integration system which focuses on the information of computer science teachers in Chinese universities. The information from heterogeneous sources are automatically extracted and re-organized into structured format. The system consists of four modules. The web page retrieval module obtain web pages with the help of topical crawler within fixed websites. In addition, search engine is also used to retrieve web pages and a classifier is then employed to detect informative pages. The second module is web page structure classification module. It combines rule-based method and machine learning based method to classify web pages into five categories according to the page layout and content. In information

extraction module, they extract teachers' attributes from web pages based on characteristics of each page category. Finally they add an information updating module to ensure information is the latest.

4.4 OfCourse: Web Content Discovery, Classification and Information Extraction for Online Course Materials

In [44], they create a vertical search engine for university courses. They first use a focused crawler to retrieve course related pages from university websites. Combined with the crawler is a navigational rank algorithm which makes the crawling process more efficient and the retrieved pages more relevant. After retrieving all relevant pages, they employ a joint statistical model to perform web classification and web information extraction together as they find out in the university course domain, the results of the aforementioned two tasks may be interdependent with each other. Finally course metadata such as title, ID and time are extracted from over 60,000 courses from top 50 schools in the US. There are three features provided with the course search engine: basic keyword search, advanced search, open framework for online courses. The last feature will allow users to add a new course to the portal.

4.5 WINACS: Construction and Analysis of Web-based Computer Science Information Networks

In [45], they try to construct an information network in computer science. Their work includes three modules: web structure mining, information network analysis, and advanced query processing. In the web structure mining module, they first propose a hybrid approach for general list discovery on the web and then use the idea of parallel paths to find similarly typed entities. Finally they make use of anchors found in the link paths to semantically define the entities. In the information network analysis module, they first do a ranking-based clustering on different types of entities and hierarchical network structure analysis and then perform query-based information network extraction and analysis. Finally they deal with issues regarding link-based object resolution and disambiguation for bibliographic networks. In the advanced query processing module, they provide integrated search functionality and promotion query analysis.

4.6 ArnetMiner: Extraction and Mining of Academic Social Networks

In [46], they develop a social network system called ArnetMiner. They address four major problems in extracting and mining an academic social network: 1) they propose a unified approach to do researcher profiling using a conditional random field model. 2) they propose a name disambiguation algorithm and use it to integrate researcher information with publications from digital libraries. 3) they propose a new model called Author-Conference-Topic to tackle the expertise search problem. 4) they make

an attempt to find connections between researchers using shortest path search and depth-first search to find top K ranked results.

4.7 Two Major Works

There are two major works that have inspired and motivated our own work. The first one is WINACS and the second one is ArnetMiner, both of which have been discussed in the last section. In this section, we talk about these two works in various technical aspects and give a summary of them.

4.7.1 Overview

First of all, both works share the same goal of building an academic information network. Second, they both combine web information extraction techniques and database technologies to realize the common goal. The grand goal of building a fully functioning academic information network consists of several subgoals, which include extracting academic institutions (and the researchers in them), integrating with publication information (probably from digital library sources), and providing expertise search and researcher association search functionalities. Both works have had substantial effort on these subgoals. The following is a summary of the commonalities of the two works.

Domain Both works focus on the academic domain.

Market Both works aim to provide semantic search functions.

Technique Both works take advantage of information extraction techniques.

4.7.2 Web Page Retrieval

In ArnetMiner, they assume that all researcher names are given when performing the researcher profiling task. They first utilize the Google API to get a list of relevant documents and then employ a classification model to identify whether or not a document in the list is really "related" to the researcher. Their approach has several limitations. For example, in our case we do not know the name of the researcher that we want to extract in advance. This is usually true when we do not have existing databases that hold such information from previous work. Another problem is due to the capability of the Google search engine. That is, if the person is not well known it might not be able to get decent Google hits. One more inherent limitation with traditional search engines is that they fail to analyze the query semantically. Considering the name "Charlene Song", Google is likely to return results related to the song "I've never been to me" by Charlene. Even considering people who are really famous, there are also problems with Google search. For example, Shafi Goldwasser, who is rewarded the prestigious Turing Award recently, does not have her homepage well indexed by Google.

In WINACS, they also make a similar assumption that at least a faculty member's homepage is given. Given the school homepage and one faculty member's homepage, they try to extract other faculty members by exploiting parallel paths. First, they have to find the shortest link path between the school homepage and the faculty member's homepage by traversing the school site. This step is computationally expensive and time-consuming if practical at all. Then they try to find all parallel paths that lead to other faculty members by discovering general lists. Finally, they have to semantically define extracted faculty information using anchor text found in the link paths.

Following is a summary of both works in terms of web page retrieval, which are explained in three dimensions:

Scalability Both works assume that names of professors are available in some existing database or known in advance. This is not always possible. Even if it is possible, when we want to expand our database of professors we have to collect more names manually. This makes the system not scalable.

Accuracy In ArnetMiner, they use Google to retrieve professor's homepage given his or her name. While this approach works with well known or accomplished professors, both precision and recall might drop when the professors are not so famous. This approach is actually flawed given the fact that quite a portion of professors around the world have little visibility. What is more, although ArnetMiner has achieved decent results on the researcher profiling problem, the results can potentially be improved by incorporating context information from the academic unit.

Efficiency WINACS proposes a semantic mining algorithm by working backward from a professor's homepage to the school homepage. They manage to find semantic information including a professor's school and department through finding the shortest path from the school homepage to the professor's homepage. The school homepage and professor's homepage must be known in advance. Then they use the idea of parallel paths to find other professors in the same school and other schools. This approach depends heavily on the outcome of the list discovery algorithm, for which they adopt a simple visual-based one. Another problem is that their program has to search through the site up to a depth of five or more, which can be computationally inefficient.

4.7.3 Data Complexity

ArnetMiner only deals with information extraction from researcher's homepages, ignoring information from the organizational context. In particular, their work focuses on extracting a relation of k-tuple while our work aims to extract a complex object with hierarchically organized data. In WINACS, they only extract information from a faculty list and do not mention information extraction from the faculty member's homepage. Their organizational context is limited to the school or department level.

4.7.4 Summary

Table 7: Two Major Systems

	Artminer	**WINACS**
Discipline	Computer Science	Computer Science
Extraction Level	Individual level	Sub-organizational level
Web Page Retrieval	Google search	Backward navigation
Automation Degree	Manual annotation of training examples	Full automation

Chapter 5

Implementation

In this chapter we first give an overview of our overall approach and explain decision making rationales and then cover the implementation details of each component of our extraction system. In particular, we first articulate the algorithms used to extract various data from a single candidate page and then show how to retrieve these candidate pages and integrate data extracted from different pages.

5.1 Three Principles

We do data extraction based on the following three principles:

1. The system must be fully automatic without human intervention.
2. The precision of the extracted data must be as high as possible.
3. We try to improve the recall when the precision is high enough.

These principles are outlined according to their priorities. First of all, we intend the system to be fully automatic based on two beliefs. On one hand, we are able to generalize patterns from target web pages, which makes automatic extraction possible. On the other hand, we aim to extract hundreds of Canadian and US university websites, which makes it impractical to use manual or semi-automatic methods. Second,

we try to ensure high accuracy of the extracted data and we even do so at the cost of missing some data. Data accuracy is an important measure of the utility of our system. Finally, we treat recall improving as a bug-fixing or patching process. We identify missing data and devise solutions for each case.

5.2 Information Sources

Our primary source of input is the official website of each university. Information on divisions, units and faculty members is all to be extracted from university websites. For university general information, it is hard to locate the information pages from university websites. We can employ search engines to retrieve relevant pages, but to extract information from various retrieved pages is not a trivial task. We decide to extract university general information from its corresponding Wikipedia page.

5.3 Ontology-based Extraction

We use the term divisions to refer to the upper-level classification of academic units and the term units to refer to the lower-level classification of academic units. In terms of specific academic units, divisions can be colleges, divisions and schools while units can be schools and departments. Unless specified otherwise, we keep using the two terms in the general sense. The following three ontological relationships are considered during extraction.

First Divisions → Divisions → Units → Faculty Members.

Second Divisions → Units → Faculty Members.

Third Units → Faculty Members.

Since units are always at the bottom of the relationships, we only extract faculty member information from units even if upper-level divisions may have a big list of faculty members from all affiliated units. The second relationship accounts for the vast majority of our extraction tasks. In other words, most universities are first divided into several divisions and these divisions are in turn divided into units. One typical example of the first relationship is universities that have College of Arts and Science. Instead of being directly divided into units, College of Arts and Science is often divided into smaller divisions first. Universities that only have a big list of departments fall into the third category. A relatively complete example of a university ontology is available in [47].

5.4 Three-staged Process

We divide the entire extraction process into three stages. In the first stage, we retrieve all possible web pages that may contain desired information. In the second stage, we extract desired information from each individual page. In the last stage, we try to integrate information extracted from different pages based on some criteria.

5.4.1 Page Retrieval

A university's website can easily contain thousands of pages, but not all pages are of interest to the extraction task. Most pages are not relevant for a domain-specific extraction. We use a focused web crawler to collect candidate web pages for extraction and incorporate page selection heuristics to speed up the crawling process. Our web crawler obeys the following rules.

Restricted Domain The web crawler only collects pages within a specified university domain. For example, only pages under "carleton.ca" will be considered for Carleton University.

File Extension Resources with certain file extensions will not be visited. These resources are mainly media files including .doc, .xls, .ppt, .mp3, .jpg, .avi and many others.

Crawling Depth For all three extraction tasks including division list extraction, unit list extraction and faculty list extraction, we assume that the crawling depth is 3. In other words, we only consider pages of depth no more than 3 for division list extraction starting from university homepage, for unit list extraction starting from division homepage and for faculty list extraction starting from unit homepage. This assumption makes our crawler practical and is true as far as our investigation goes.

Visited Pages The crawler remembers already visited pages to ensure that they will not be visited again.

Politeness An http request will only be made every other second within the same domain.

Multi-threaded We adopt one thread per university scheme.

Page Selection Heuristics

Despite the rules to obey, the web crawler often needs to collect hundreds of candidate pages for each extraction task. It can be a huge burden for both the crawler and the extraction program. As an experienced web user, we are able to navigate through websites and find relevant pages quickly. This is because a website is usually designed to be easy for human users to browse. For domain websites, the site designers use common domain keywords consistently to help users find information. For example, we look for keywords such as “faculty” and “people” when we want to browse professor information in an academic unit. We incorporate such page selection heuristics into

our crawler to reduce the number of candidate pages. In particular, we first process links that contain related domain keywords. The total number of candidate pages retrieved this way is usually less than 100, which is substantial improvement over link traversal without selection. Only when we are not able to extract desired information using page selection heuristics, we need to traverse all possible links.

5.4.2 Information Extraction

Since information extraction is the main stage of the entire process, we discuss it in detail. After retrieving relevant pages, we adopt different approaches for different extraction tasks. For division and unit information extraction, we utilize visual features of web pages [48] while for faculty information extraction, we make use of HTML encodings of the pages.

Simple List and Complex List

Most division lists and unit lists are considered as simple lists. The single piece of information is division name or unit name (with a likely underlying hyperlink). Most faculty lists are considered as complex lists since they can contain multiple pieces of faculty information (i.e., various faculty attributes). In real-world cases, some division lists and unit lists can contain multiple pieces of information (e.g., some basic information about the division and unit or its affiliated units and programs as a sublist) and thus can be regarded as complex lists while some faculty lists can contain only one piece of information (i.e., the faculty member's name and a likely underlying hyperlink) and thus can be regarded as simple lists. For our extraction purposes, we treat every division and unit list as a simple list and every faculty list as a complex list in a unified way. In other words, when we look at a division or unit list we only focus on the division or unit name and ignore other information if there is any and when we look at a faculty list we see each item as an information block or

a data record that consists of multiple pieces of information although it can just be a single faculty name. This way we are able to take advantage of different algorithms for different kinds of lists to achieve simplicity and improve accuracy.

DOM-based Approach and Visual-based Approach

A DOM-based approach is based on the HTML encodings of web pages. By exploiting the DOM structure of the page, we are able to find patterns that facilitate our extraction. The visual-based approach is based on the visual representation of web pages [48]. In terms of information extraction, we can make use of various visual features of each element on web pages to cluster and group page elements. In particular, two types of visual features can be used in web list extraction:

Position features x- and y-coordinates, height and width of the element.

Style features Font, color, background color, border properties of the element.

The visual-based approach is based on the hypothesis that there exist distinct visual features for data records and data items within each record. Our observation based on a large number of real-world web pages is consistent with this hypothesis. The same hypothesis is also made and justified in [17]. With DOM-based approach we do not need to render the web page while with visual-based approach we need to retrieve the rendering information of each HTML element. A HTML parser such as JSoup is good enough for the DOM-based approach; however, a web page rendering engine such as those seen in modern browsers is required for the visual-based approach. In our implementation, we use a lightweight rendering engine called CSSBox written in pure Java. Rendering a web page can potentially cause performance issues. Depending on the tool used for rendering, some web pages cannot be rendered or rendered properly due to various problems. For example, the CSSBox library does not support Javascript. However, the visual-based approach usually has better performance in

terms of recall and precision than the DOM-based approach because HTML tags are eventually used for rendering purposes. For division list and unit list extraction, we use visual information to generate candidate lists on web pages. For faculty list extraction, we base our algorithm on DOM features. In particular, we employ pattern mining techniques to generate faculty list candidates. We do not use a visual-based approach for faculty list extraction because we treat all faculty lists equally as complex lists, but unfortunately not every piece of information in a complex list is visually aligned.

Free Text Extraction

Extracting information from a faculty member's homepage is considered as free text extraction. Although each faculty member's homepage often shares a common template within units, different units can have quite different templates. Given the extraction scale of our problem, it is not practical to use techniques based on template discovery. We treat every homepage to be extracted as a free text document. However, it is a little different from pure natural language text, which conforms to English grammars and has no HTML tag encodings. Thus, conventional NLP techniques are not suitable for homepage extraction task. We employ rule-based algorithms to extract basic faculty information from their homepages. We design extraction rules with domain knowledge from an expert and take advantage of regular expressions to extract specific information. Although they achieve good results using corpus-based approach on faculty homepages in [49], we do not use it in the spirit of full automation. In their corpus-based approach, they need to manually annotate a lot of examples for training purposes.

5.4.3 Information Integration

When we extract information from multiple sources, we need to integrate all extracted information to obtain the desired result. For example, we can often get division information from more than one candidate page. Thus, we need a mechanism to identify the desired information. We use three pieces of information to integrate data extracted from more than one source.

List Content In division lists, we can see keywords such as "Faculty", "College", "Division" and "School". In unit lists, we can see keywords such as "School" and "Department". In faculty lists, we can see keywords such as "Professor" and "Instructor".

List Heading There is often a heading for each web list. Figure 21 shows the division (faculty) list of Carleton University. We can see that the heading of this list is "Faculties".

Link Anchor To arrive at the division list page in Figure 21, we start from Carleton's homepage, click the link with anchor text "Academics" and then the link with anchor text "FACULTIES". Both "Academics" and "FACULTIES" are link anchors on the path leading to the list page.

We use these three pieces of information to make priority rules, based on which we integrate information extracted from multiple sources.

Figure 21: Faculty List of Carleton University

5.5 Assumptions, Tools and Disclaimers

In this section, we first discuss assumptions made in designing the entire system. We then introduce the tools including programming languages and libraries we use in implementing the system. Last but not least, we summarize the limitations of our system.

5.5.1 Assumptions

Since our division, unit and faculty information extractions all depend on the presence of a general web list, we assume that there are at least two items in the list. That means, for example, if a division has only one unit or a unit has only one faculty member, our system is not able to handle that. Our second assumption is that all division lists and unit lists reside within one single page. This assumption is true for the universities we examine. We make this assumption to simplify the system design. The third assumption we make is that the maximum number of classification levels for academic units is 3. In other words, the academic unit hierarchy within a university has a maximum depth of 3, which holds as far as our investigation goes. Finally,

we assume that division names and unit names are visually aligned in some way and these names share the same visual features including fonts, heights, colors and borders and that faculty member information is encoded using the same or similar tags in a continuous region for all faculty members in the same list. This last assumption is the foundation of our list extraction algorithms.

5.5.2 Tools

Our system is implemented in pure Java so it can run on any Java-supported platform. We use the Jsoup library to parse web pages into DOM objects and the CSSBox library to retrieve visual information about web pages.

5.5.3 Disclaimers

First of all, any cases where the four assumptions are violated are not handled by our system. Second, our system is not capable of handling Javascript-related problems, which means 1) we are not able to process redirect links produced by Javascript and 2) we are not able to extract page content generated by Javascript. At the moment, we cannot find a good Java library that supports Javascript.

5.6 Division List and Unit List Candidate Generation

In chapter 3, we have defined four kinds of division and unit list. They are nested list, indexed list, tiled list and horizontal list. In our implementation, we extend the classification by combining features of different lists to achieve better results. In particular, any division list and unit list fall into one of the following seven categories:

Vertical List List items share the same x-coordinate.

Approximate Indexed List It is a sub-list of a bigger vertical list where adjacent items are not equally far apart.

Strict Indexed List It is a sub-list of a bigger vertical list where adjacent items are equally far apart.

Horizontal List List items share the same y-coordinate.

Tiled List It consists of multiple rows and columns where items in the same column share the same x-coordinate and items in the first row share the same y-coordinate.

Nested List It is a special kind of vertical list where adjacent items are farther apart than in vertical list.

Nested Tiled List It is a special kind of tiled list where adjacent items in the same column are farther apart than in tiled list.

We first introduce data structures used in our algorithms and then articulate why we should and how we can generate division and unit list candidates based on these seven categories.

5.6.1 Data Structures and Preprocessing

In our implementation we take advantage of the Java CSSBox [8] library to retrieve visual information of the candidate page. We create a class called NodeInfo whose instances hold information about text nodes of a rendered web page. The specification of NodeInfo class is given in Table 8.

Table 8: Specification of NodeInfo Class

Field Name	Description
text	the text content of the node
url	the underlying hyperlink of the text if it is encoded by the a tag
x	the x-coordinate of the text on the page
y	the y-coordinate of the text on the page
height	the content height of the text
index	the group number of an indexed list from a bigger vertical list
position	the index of the NodeInfo object in the list that holds all NodeInfo objects corresponding to the page
style	including font family, font size, font weight, borders, colors

As a preprocessing for all the following seven algorithms, we retrieve all text nodes from the candidate page and store the retrieved information including text, url, x, y, height and style for each text node in a NodeInfo object. Image nodes are retrieved as well since some division and unit information is encoded using images. For image nodes, we treat them the same way as normal text nodes except that we set the text value to the image's alt or title attribute (when alt is absent, we use the title attribute). This is a simple preprocessing which actually maps the web page to a list of NodeInfo objects.

5.6.2 Generating Vertical List Candidates

In Figures 22, 23 and 24, all division names are vertically aligned and these division names have exactly the same appearance. By the same appearance, we mean all names share the same text height, the same font size, the same font family, the same

font color, the same font weight, the same background color and the same border around the names. However, the underlying encoding tags for names from different lists can be very different. Names from the same list can be encoded by a variety of tags including the li tag, the p tag, the div tag, the tr tag and the span tag. Even within the same list, some name may be encoded by different sequences of tags. We have seen examples where the first item is encoded differently although it looks no different from others. In order to extract all division names without any irrelevant items, we need to group them together somehow. We base our algorithms on these aforementioned visual features.

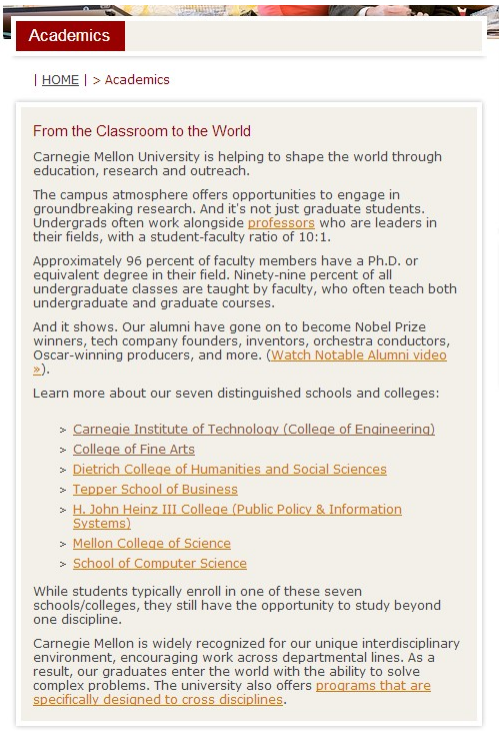


Figure 22: Dealing with vertical list one

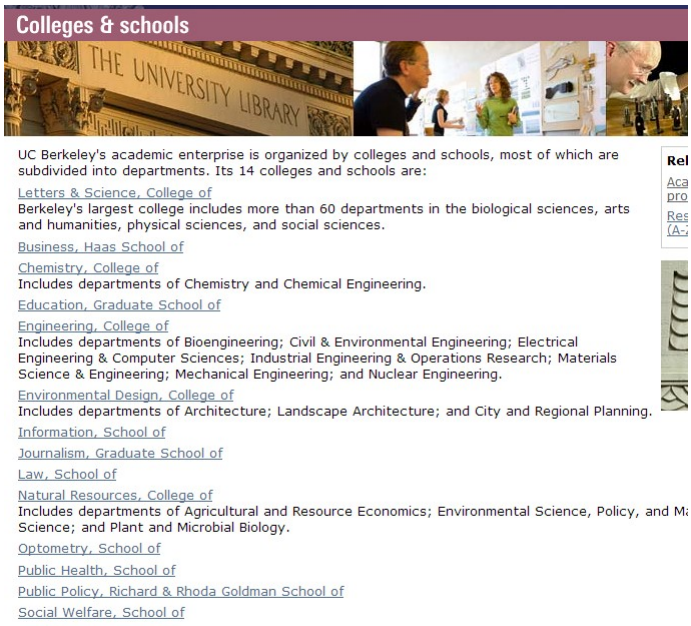


Figure 23: Dealing with vertical list two

Figure 24: Dealing with vertical list three

As the first way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their text content height and their style. A vertical list captures the situation where the list consists of all items that share the same visual appearance and are vertically aligned. The detailed algorithm is given in Algorithm 5.1.

Algorithm 5.1: Generating Vertical List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1 Cluster the NodeInfo objects from input according to the combination of x,
  height, style to get a clustering (C_1, C_2, ..., C_m);
2 for i ← 1 to n do
3 |   if C_i has fewer than 2 NodeInfo objects then
4 |   |   remove C_i from the clustering;
5 |   end
6 end
7 return the updated clustering;
```

5.6.3 Generating Approximate Indexed List Candidates

In Figure 25, 26 and 27, the division lists are not the entire vertical lists but only part of the entire vertical lists. The previous algorithm for vertical list candidates is not able to separate the division lists from the remaining part of the entire vertical lists. For example, the vertical list algorithm will group both divisions and institutes in one single list in Figure 25, which is undesirable. Thus, we need extra effort to break the entire vertical list into semantically independent sub-lists. First of all, we can use the headings and the hr tag as a separator. However, these separators can be missing in some cases, so separator alone is not enough. Another important observation is that the vertical distance between two lists is much bigger than that between adjacent items in the same list. We do not require that adjacent items in the same list are

equally far apart. In fact, if we require so, the school lists in both Figure 26 and 27 will be incorrectly broken down since the adjacent items in the division list are not equally far apart. This is also why we name this algorithm "approximate indexed".

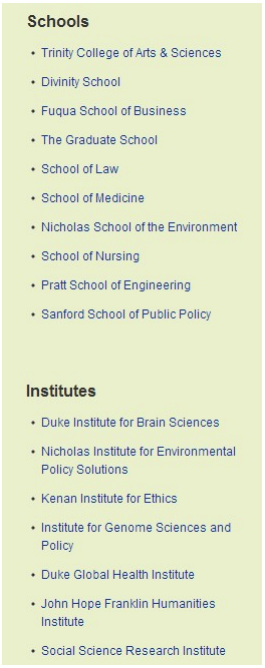


Figure 25: Dealing with approximate indexed list one

Figure 26: Dealing with approximate indexed list two

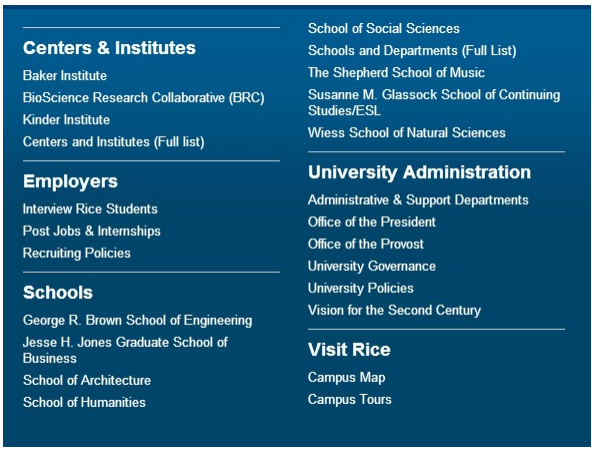


Figure 27: Dealing with approximate indexed list three

As the second way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their text content height, their style, their group index and their y-coordinate. An approximate indexed list captures the situation where the list is part of a bigger vertical list and the distance between the list and other lists is bigger than the distance between items of the list. In our implementation, we require that the distance between two different lists be two times bigger than that between adjacent items in the same list. The threshold two is set based on experimental results. By incorporating the index information we further break down a big vertical list into sublists, which are put into different clusters instead of the same cluster. See Figure 8 on page 30 for an illustration. Every item in the division list is assigned group index 1 while every item in the list below is assigned group index 2. We break the big vertical list into two and group them differently. The big visual gap in between makes them apparently two different lists. The detailed algorithm is given in Algorithm 5.2.

Algorithm 5.2: Generating Approximate Indexed List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1  Cluster the NodeInfo objects from input according to the combination of x,
   height, style to get a clustering (C_1, C_2, ..., C_m);
2  for i ← 1 to n do
3      if C_i has fewer than 2 NodeInfo objects then
4          remove C_i from the clustering;
5      else
6          sort C_i according to the y value of each NodeInfo object;
7          compute the y differences of all adjacent NodeInfo objects and store
           them in array interval;
8          currentIndex = 0;
9          set the index values of the first two NodeInfo objects in C_i to
           currentIndex;
10         foreach nodeinfo in C_i excluding the first two do
11             if the y difference of nodeinfo and its previous object is two times
               bigger than the previous difference in interval then
12                 increment currentIndex;
13                 set the index value of nodeinfo to currentIndex;
14             else
15                 set the index value of nodeinfo to currentIndex;
16             end
17         end
18     end
19 end
20 recluster all NodeInfo objects remaining in the original clustering according to
   the combination of x, height, style, index;
21 return the new clustering;
```

5.6.4 Generating Strict Indexed List Candidates

In Figure 28 and 29, the division lists are very similar to those from previous section. However, there is one subtle difference between them, which makes the previously proposed algorithm fail to separate the division lists. The difference is that in both Figure 28 and 29 the distance between different lists is not significantly bigger than that between adjacent items in the same list. Using the previous algorithm, we are not able to separate the division lists from the bigger vertical list since the distance between two lists is no more than two times bigger than that between adjacent items in the same list. We also observe that in such cases where two lists are so closely placed, items in the same list are usually equally far apart. Thus, we slightly modify the previous algorithm to deal with such cases.

Figure 28: Dealing with strict indexed list one

Figure 29: Dealing with strict indexed list two

As the third way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their encoding tag, their parent's encoding tag, their text content height, their group index and their y-coordinate. A strict indexed list captures almost the same situation as an approximate indexed list except that we require adjacent items in the same list be equally far apart from each other. The detailed algorithm is given in Algorithm 5.3.

5.6.5 Generating Horizontal List Candidates

In Figure 30, we can see that the unit names are horizontally aligned, which is very different from the vertical list. Another example has been seen in Figure 9 on page 33. The algorithm for vertical list is not able to handle such cases. However, the good thing is that all names in the list have the same appearance as well. In addition, they share the same y-coordinate. Now we give the algorithm for grouping such kind of unit names.

Algorithm 5.3: Generating Strict Indexed List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1  Cluster the NodeInfo objects from input according to the combination of x,
   height, style to get a clustering (C_1, C_2, ..., C_m);
2  for i ← 1 to n do
3      if C_i has fewer than 2 NodeInfo objects then
4          remove C_i from the clustering;
5      else
6          sort C_i according to the y value of each NodeInfo object;
7          compute the y differences of all adjacent NodeInfo objects and store
           them in array interval;
8          currentIndex = 0;
9          set the index values of the first two NodeInfo objects in C_i to
           currentIndex;
10         foreach nodeinfo in C_i excluding the first two do
11             if the y difference of nodeinfo and its previous object is not equal to
               the previous difference in interval then
12                 increment currentIndex;
13                 set the index value of nodeinfo to currentIndex;
14             else
15                 set the index value of nodeinfo to currentIndex;
16             end
17         end
18     end
19 end
20 recluster all NodeInfo objects remaining in the original clustering according to
   the combination of x, height, style, index;
21 return the new clustering;
```

Figure 30: Dealing with horizontal list one

As the fourth way to generate candidate lists, we cluster the text nodes according to the combination of their text content height, their style and their y-coordinate. A horizontal list captures the situation where the list consists of all items that share the same visual appearance and are horizontally aligned. The detailed algorithm is given in Algorithm 5.4.

Algorithm 5.4: Generating Horizontal List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1 Cluster the NodeInfo objects from input according to the combination of y,
  height, style to get a clustering (C_1, C_2, ..., C_m);
2 for i ← 1 to n do
3 |   if C_i has fewer than 2 NodeInfo objects then
4 |   |   remove C_i from the clustering;
5 |   end
6 end
7 return the updated clustering;
```

5.6.6 Generating Tiled List Candidates

In Figure 31 and 32, we can see that both division lists consist of multiple rows and multiple columns. Neither of the vertical and horizontal list algorithms is able to handle such cases. Although we can use the vertical list algorithm to get two individual lists of divisions, we need extra effort to group the two individual lists as a whole. We observe that the first item of each column shares the same y-coordinate, which we will use to merge multiple individual lists into a single list. However, this condition is not sufficient enough yet. In Figure 32, we can see the rightmost vertical list is also horizontally aligned with the actual division list. We can see another example in Figure 33 where an irrelevant vertical list is horizontally aligned with the actual division list. Thus, we need more effort to ensure no irrelevant list be merged. We observe that there are headings named "Related Links" and "Academic Departments" respectively to semantically separate the two unrelated lists. So, when we merge two lists , we first check whether there is a heading in between. If yes, we do not merge them. Now we give the algorithm for merging such individual lists into a whole.

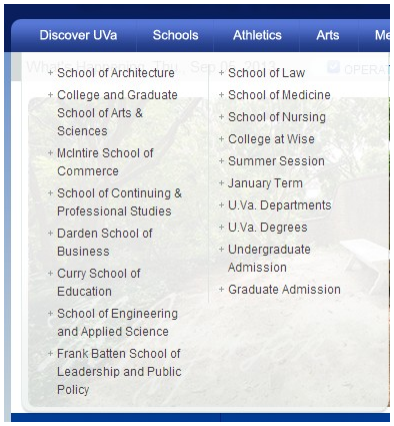


Figure 31: Dealing with tiled list one

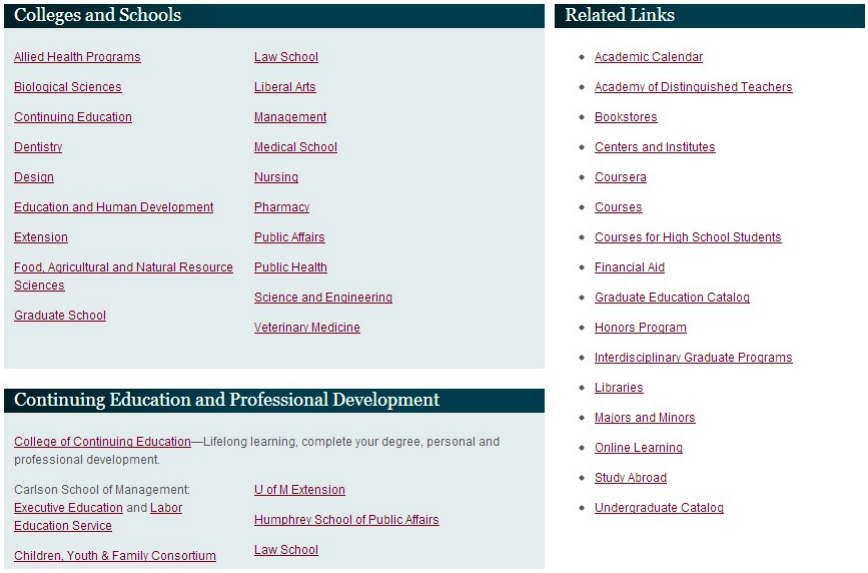


Figure 32: Dealing with tiled list two

Figure 33: Dealing with tiled list three

As the fifth way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their text content height, their style, their

group number and their y-coordinate. A tiled list captures the situation where the list consists of multiple columns, each of which is in fact an approximate indexed list and the first rows of which are horizontally aligned. The detailed algorithm is given in Algorithm 5.5.

5.6.7 Generating Nested List Candidates

In Figure 34, 35 and 36, these division lists are quite different from previously seen examples in that they contain a lot more information than just the division names. In particular, under each division name there is a sub-list which contains its affiliated units, general division information, or both. Thus, we call such division list nested list. In fact, the algorithm for vertical list is able to extract division names from such nested lists. However, since some unit lists are on the same page as the division list (as its sub-lists), chances are that one unit list can be mistakenly extracted as a division list or the result can be a mix of both division list and unit list. Such results are not acceptable. What is more, since division list and unit list are at two different semantic levels, the algorithm should better be able to distinguish the main list from the sub-lists. There are various features we can use to distinguish them. First of all, adjacent division names are usually far apart from each other. Second, the division names typically have heavier font weight than unit names. Third, the text height of the division name is often bigger than that of unit names. Finally, division names usually have smaller x-coordinates than unit names. In other words, they are closer to the left side.

Algorithm 5.5: Generating Tiled List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1  Cluster the NodeInfo objects from input according to the combination of x,
   height, style to get a clustering (C_1, C_2, ..., C_m);
2  for i ← 1 to n do
3      if C_i has fewer than 2 NodeInfo objects then
4          remove C_i from the clustering;
5      else
6          sort C_i according to the y value of each NodeInfo object;
7          compute the y differences of all adjacent NodeInfo ojbects and store
           them in array interval;
8          currentIndex = 0;
9          set the index values of the first two NodeInfo objects in C_i to
           currentIndex;
10         foreach nodeinfo in C_i excluding the first two do
11             if the y difference of nodeinfo and its previous object is two times
               bigger than the previous difference in interval then
12                 increment currentIndex;
13                 set the index value of nodeinfo to currentIndex;
14             else
15                 set the index value of nodeinfo to currentIndex;
16             end
17         end
18     end
19 end
20 recluster all NodeInfo objects remaining in the original clustering according to
   the combination of x, height, style and index;
21 sort every list in the new clustering by the y value of NodeInfo object;
22 foreach list L in the new clustering do
23     if there exists at least one other list whose first item has the same y value
       as that of L and the position value of the last NodeInfo object in one list is
       exactly one smaller than that of the first NodeInfo object in another list
       then
24         merge all these lists with the same y value into one;
25     else
26         remove the list L from the clustering;
27     end
28 end
29 return the merged clustering;
```

Figure 34: Dealing with nested list one

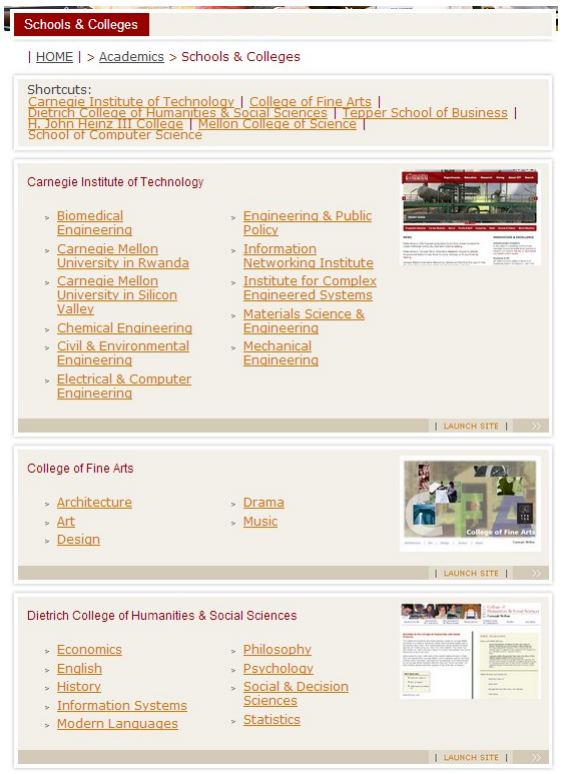


Figure 35: Dealing with nested list two

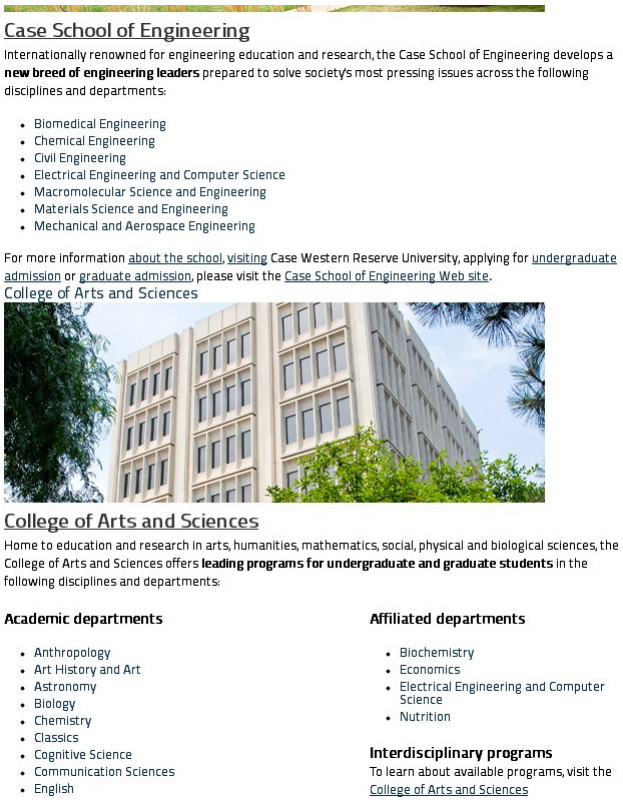


Figure 36: Dealing with nested list three

As the sixth way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their text content height and their style. A nested list captures the situation where the list consists of only one column and the vertical distance between adjacent items is relatively big. We set the distance to be no smaller than 50 px. In the case of nested lists, we do not do grouping because we assume the gap between adjacent items is at least 50 px and there is little chance of another list existing with the same x-coordinate. The detailed algorithm is given in Algorithm 5.6.

5.6.8 Generating Nested Tiled List Candidates

In Figure 37 and 38, the two division lists are both tiled lists and nested lists. The division list in Figure 37 has hidden sub-lists of programs, for which we can use the

Algorithm 5.6: Generating Nested List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1  Cluster the NodeInfo objects from input according to the combination of x,
   height, style to get a clustering (C_1, C_2, ..., C_m);
2  for i ← 1 to n do
3  |  if C_i has fewer than 2 NodeInfo objects then
4  |  |  remove C_i from the clustering;
5  |  else
6  |  |  sort C_i according to the y value of each NodeInfo object;
7  |  |  if Any adjacent NodeInfo objects in C_i have y value difference smaller
   |  |  than 50 then
8  |  |  |  remove C_i from the clustering;
9  |  |  end
10 |  end
11 end
12 return the updated clustering;
```

algorithm for nested list. For the list in Figure 38, it does not have any sub-lists, however, we still treat each column of the list as a nested list. Their adjacent items are far apart, so such kinds of list can potentially have sub-lists of departments or programs. Since the nested list algorithm or the tiled list algorithm alone is not able to handle such cases, we need extra effort to merge the results of the nested list algorithm in a similar way to the tiled list algorithm. The only difference is that in the tiled list algorithm we use headings to separate irrelevant lists; however, in the nested tiled list algorithm we only need to consider the horizontal alignment of the first item of each column. The observation is that there is no chance that other irrelevant lists can be horizontally aligned with the division list. What is more, the division names are usually encoded using headings.

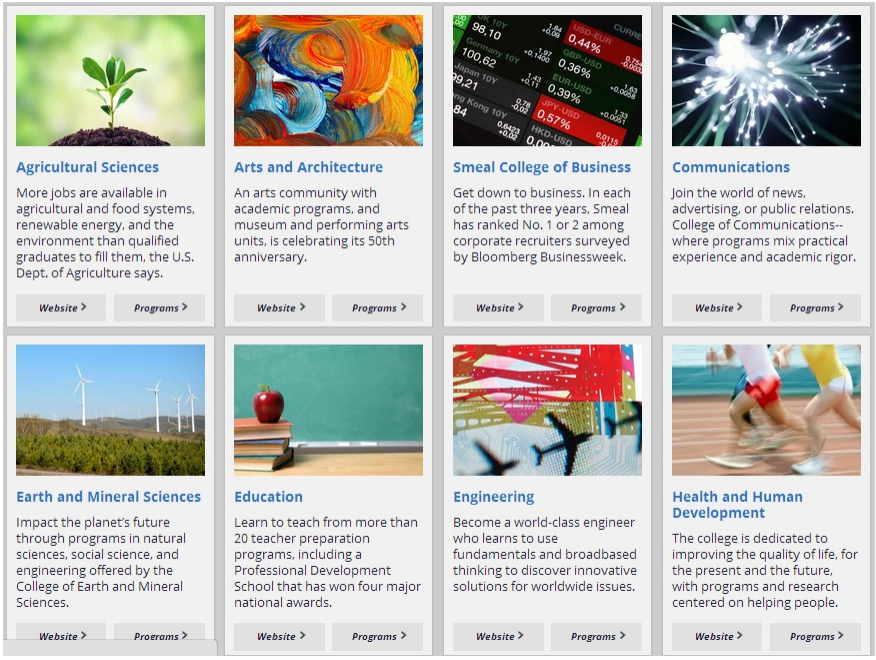


Figure 37: Dealing with nested tiled list one

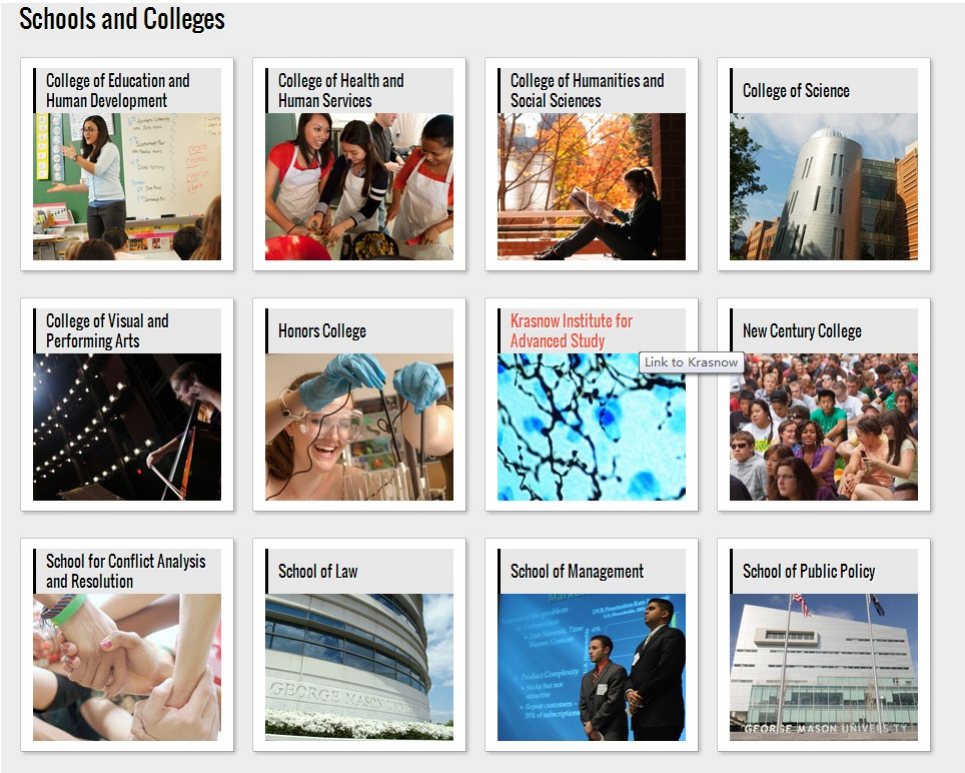


Figure 38: Dealing with nested tiled list two

As the seventh way to generate candidate lists, we cluster the text nodes according to the combination of their x-coordinate, their text content height, their style and their y-coordinate. A nested tiled list captures the situation where the list consists of multiple columns and the vertical distance between adjacent rows is relatively big. We set the distance to be no smaller than 50 px in our algorithm. The detailed algorithm is given in Algorithm 5.7.

Algorithm 5.7: Generating Nested Tiled List Candidates

input : A list of NodeInfo objects corresponding to a candidate list page
output: A Clustering $(C_1, C_2, ..., C_n)$ where each element C_i is a list of NodeInfo objects

```
1  Cluster the NodeInfo objects from input according to the combination of x,
   height, style to get a clustering (C_1, C_2, ..., C_m);
2  for i ← 1 to n do
3      if C_i has fewer than 2 NodeInfo objects then
4          remove C_i from the clustering;
5      else
6          sort C_i according to the y value of each NodeInfo object;
7          if Any adjacent NodeInfo objects in C_i have y value difference smaller
           than 50 then
8              remove C_i from the clustering;
9          end
10     end
11 end
12 sort every list in the updated clustering by the y value of NodeInfo object;
13 foreach list L in the new clustering do
14     if there exists at least one other list whose first item has the same y value
       as that of L then
15         merge all these lists with the same y value into one;
16     else
17         remove the list L from the clustering;
18     end
19 end
20 return the merged clustering;
```

5.7 Division List Extraction

Given a candidate division list page, our goal is to extract division list from the page. The algorithm works in two steps. We first generate division list candidates using the seven algorithms and then identify the division list by utilizing a division dictionary and checking the list heading. We start with building a division dictionary.

5.7.1 Division Dictionary

We first study the content of a division name. Then we build a division dictionary by collecting keywords from division names. Table 9, 10, 11 demonstrate three different types of division lists and how we collect keywords from each division name respectively.

Table 9: Building Division Dictionary: One

Item in Division List	**Keywords Collected**
Carnegie Institute of Technology (College of Engineering)	Technology, Engineering
College of Fine Arts	Fine Arts
Dietrich College of Humanities and Social Sciences	Humanities, Social Sciences
Tepper School of Business	Business
H.John Heinz III College (Public Policy & Information Systems)	Public Policy, Information Systems
Mellon College of Science	Science
School of Computer Science	Computer Science

Table 10: Building Division Dictionary: Two

Item in Division List	**Keywords Extracted**
Letters & Science, College of	Letters, Science
Business, Haas School of	Business
Chemistry, College of	Chemistry
Education, Graduate School of	Education
Engineering, College of	Engineering
Environmental Design, College of	Environmental Design
Information, School of	Information
Journalism, Graduate School of	Journalism
Law, School of	Law
Natural Resources, College of	Natural Resources
Optometry, School of	Optometry
Public Health, School of	Public Heath
Public Policy, Richard & Rhoda Goldman School of	Public Policy
Social Welfare, School of	Social Welfare

Table 11: Building Division Dictionary: Three

Item in Division List	**Keywords Extracted**
Business	Business
Earth Sciences	Earth Sciences
Education	Education
Engineering	Engineering
Humanities & Sciences	Humanities, Sciences
Law	Law
Medicine	Medicine

We do not have firm rules of collecting keywords to build our division dictionary, but we follow two principles:

1. Instead of selecting keywords such as "School" and "College", we place our focus on keywords that describe what the division is about. We make this decision because lots of universities do not put "School"-like keywords in their division list (see Table 11 for an example).

2. We always select the most specific and yet atomic one. Consider the example in Table 9, we extract "Humanties" and "Social Sciences" from "Dietrich College of Humanities and Social Sciences". We do not pick "Humanities and Social Sciences" as a whole because we want it to be atomic or self-contained. We do not break "Social Sciences" further down to make "Social" and "Sciences" because we want it to be the most specific.

Making keywords most specific reduces the chance of other non-division name phrases containing the keywords while making keywords atomic increases the chance of other division name variations containing the keywords. We build a division dictionary of around 200 keywords by analyzing division lists from the 26 Canadian and 74 US universities. One important observation is that as we go through these 100 universities and finish the first 25 universities, we are only able to collect very few new keywords from remaining universities. This observation is important because it convinces us that our division dictionary is very comprehensive and thus very reliable.

5.7.2 Division List Identification

Now we are able to generate candidate lists in seven ways and have built a division dictionary. We need to identify the division list from these candidate lists using a number of selection criteria. In this process, only text value and url value of each NodeInfo object are used. In order for a candidate list to be identified as division list, all conditions below must be satisfied:

1. The list must have at least four items.

2. The number of words in each item's text must be no more than 10 and the text cannot contain digits.

3. At least 2/3 of the items's texts in the list contain one or more keywords from the division dictionary.

4. The item's url cannot contain negative words such as "admission", "news", "apply" and "article".

All universities we investigate have four or more divisions. We require the text in each list item have no more than 10 words based on the observation that many news article titles can hit the keywords easily. The threshold 2/3 is chosen according to empirical results. On one hand, we cannot guarantee all items in a division list hit keywords from our division dictionary. On the other hand, we need to make the threshold big enough to avoid irrelevant results. Finally, we need to check if each item's url contains negative words such as "admission", "news", "apply", "article" and "research". These negative words are often embedded in the url of each item. Given a candidate list, our identification algorithm returns true if it considers the candidate list as a division list and false otherwise. Since we only need two pieces of information of items (i.e., NodeInfo objects) in the candidate lists, we convert the list of NodeInfo objects to a list of value pairs consisting of text value and url value. The detailed algorithm is given in Algorithm 5.8.

5.7.3 Division List Priority

After applying the identification algorithm on division list candidates, we often get more than one list as the result. It is true because unit lists under each division can be on the same page as the division list and both kinds of lists can share some field keywords. See Figure 24 and 24 for two examples. Another case is that there can be two division lists of different forms on the same page. See Figure 39 for an example.

Algorithm 5.8: Identifying Division List

input : A list of value pairs $(text_1, url_1), (text_2, url_2), ..., (text_n, url_n)$
output: A boolean value

```
1  counter = 0;
2  for i ← 1 to n do
3  |   if text_i has no more than 10 words and text_i contains no digits and at least
   |   one word from the division dictionary and url_i does not contain any of the
   |   five negative words then
4  |   |   increment counter;
5  |   end
6  end
7  if counter >= n * 2/3 then
8  |   return true;
9  else
10 |   return false;
11 end
```

We define three priority rules so that only the most likely one is returned. The three rules are listed based on their priorities.

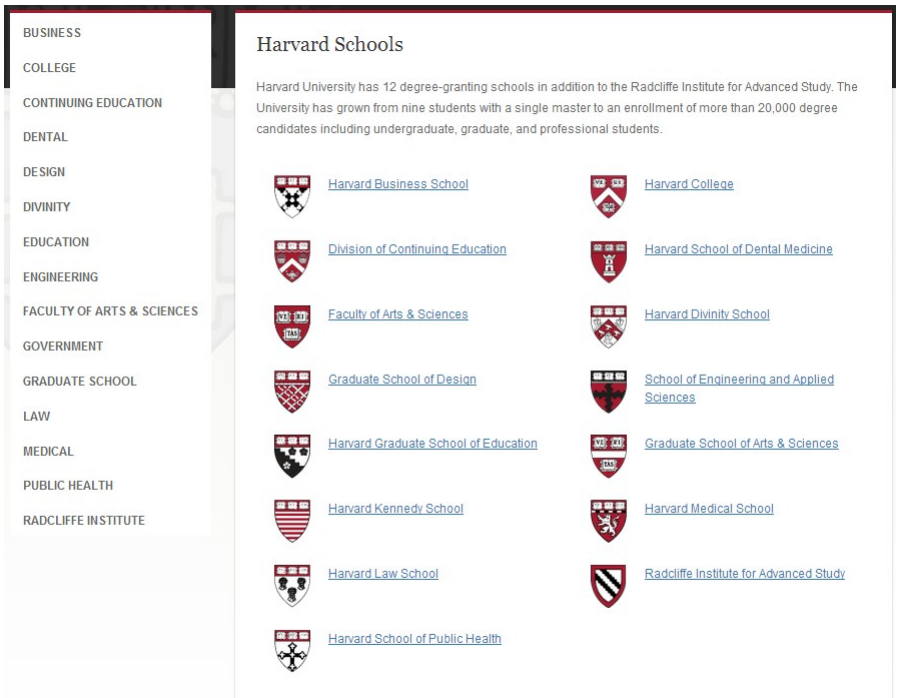


Figure 39: Two division lists of different forms on the same page

Division Name Keyword (Rule one) At least 2/3 list items contain keywords including "Faculty", "College", "Division" and "School". All these keywords are treated as case insensitive.

List Heading (Rule two) The list heading contains keywords including "Faculties", "Colleges", "Divisions" and "Schools". All these keywords are treated as case insensitive.

Candidate Generation Type (Rule three) The list generated by one of the seven candidate list generation routines can have priority over others. For example, the nested list routine has priority over the tiled list routine. Candidate lists are generated in the order of the priorities of their corresponding generation types in Algorithm 5.9.

First, we elaborate on list headings. We treat both the text just before the first list item and the HTML heading just before the first list item as list headings. Often, the two things are the same while sometimes, they can be different. Take the list in Figure 39 for an example. The text just before the first list item is a long introduction paragraph about divisions in Harvard while the HTML heading just before the first list item is "Harvard Schools" in bold face. In this case, we first check the text part for keywords and then check the HTML heading for keywords. Now we summarize the general priority rules. Lists satisfying all three rules are considered first. Then lists satisfying both rule one and rule three are considered. Lists satisfying both rule two and rule three are considered next. Finally, lists satisfying rule three are considered.

5.7.4 The Complete Algorithm

The complete algorithm works in three iterations. First we use the seven list generation routines to generate candidate lists from the division list page. Second we

use the identification algorithm to identify potential division lists from the candidate lists. Last we apply the priority rules to all potential division lists and return the most likely list. If no division lists are identified in the second iteration, we return an empty list. The complete algorithm is given in Algorithm 5.9.

In the returned list of value pairs, the text is the name of the division and the url is the homepage of the division. In the list candidate generation process, the order for the seven routines is experimentally set to achieve the best possible outcome. The general rule is that the indexed routine must come after the tiled routine and the indexed routine must come after the nested routine. If we put the indexed routine before the tiled routine, then some tiled division lists can only be partially retrieved (i.e., only one column of the tiled division list is retrieved). If we put the indexed routine before the nested routine, then some unit list can be retrieved instead of the division list (since division list can be on the same page with unit lists in a nested way).

5.7.5 Division URL Retrieval

In some division lists, the division name is not clickable, which means the division name is not encoded as a link. In that case we need extra effort to retrieve the homepage URL for each division. The URL for each division is normally located in the sibling elements of the division name element. See Figure 40, 41 and 42 for three examples. The URL retrieval algorithm works by checking all link nodes between the current division node and the next division node. If any link node contains keywords such as "Homepage" and "Website", we return the link node's url as the current division's url. The detailed algorithm for retrieving the division URL is given in Algorithm 5.10.

Algorithm 5.9: Division List Extraction

input : A web page as given by its URL
output: A list of value pairs or an empty list

```
1  Render the page using CSSBox, retrieve text, url, x, y, height, style, index
   information for each text node on the page and store the information with a
   NodeInfo object. A list L of NodeInfo objects are ready for processing;
2  divisions ← ∅;
3  candidates ← NestedTiledCandidates(L) ∪ NestedCandidates(L) ∪
   TiledCandidates(L) ∪ VerticalCandidates(L) ∪
   ApproximateIndexedCandidates(L) ∪ StrictIndexedCandidates(L) ∪
   HorizontalCandidates(L);
4  foreach list l in candidates do
5  |  if IdentifyDivisions(l) then
6  |  |  divisions ∪l;
7  |  end
8  end
9  first, second, third = null;
10 foreach list l in divisions do
11 |  if first == null and RuleOneTrue(l) and RuleTwoTrue(l) then
12 |  |  first = l;
13 |  else if second == null and RuleOneTrue(l) then
14 |  |  second = l;
15 |  else if third == null and RuleTwoTrue(l) then
16 |  |  third = l;
17 |  if first!= null and second!= null and third!= null then
18 |  |  break;
19 |  end
20 end
21 if first!= null then
22 |  return first;
23 else if second!= null then
24 |  return second;
25 else if third!= null then
26 |  return third;
27 else if divisions is not empty then
28 |  return the first list in divisions;
29 return ∅;
```

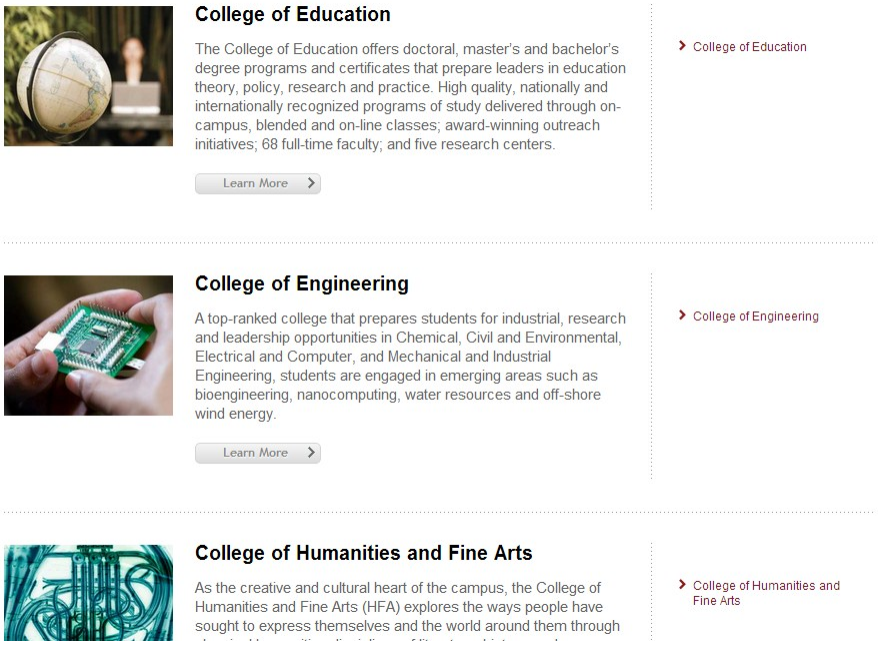


Figure 40: Separate division URL one

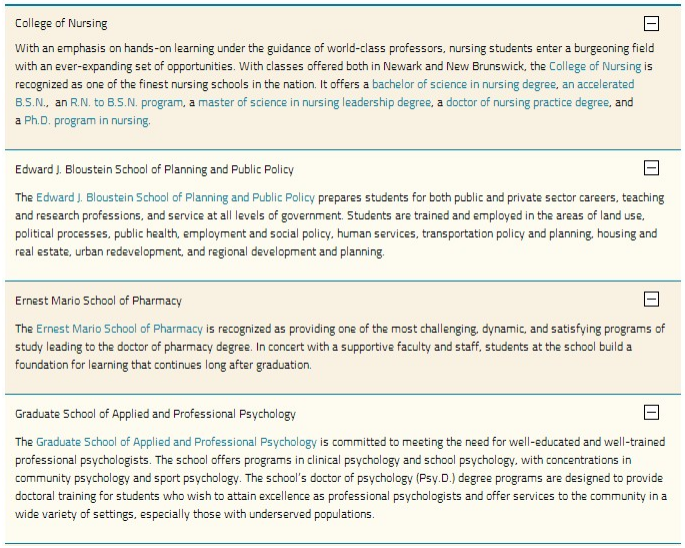


Figure 41: Separate division URL two

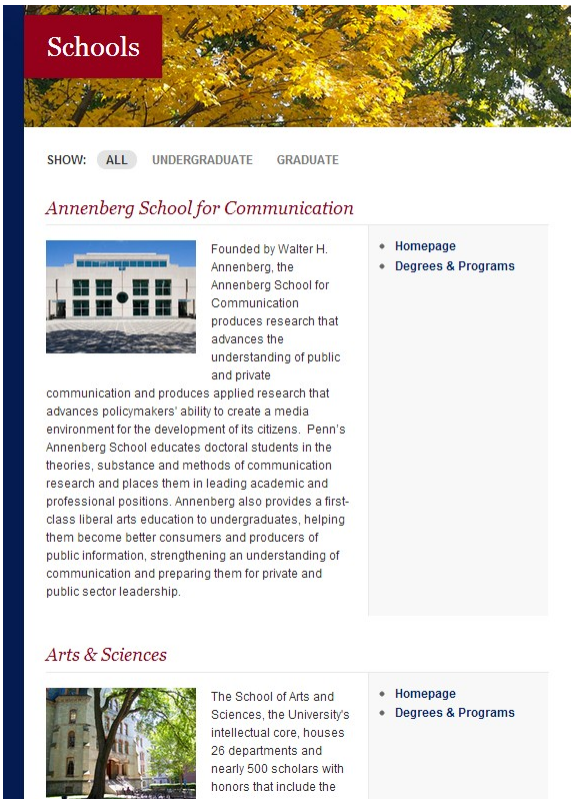


Figure 42: Separate division URL three

5.8 Unit List Extraction

Given the unit list page, the goal is to extract the unit names and homepages from it. Unit list extraction is similar to division list extraction. Like division list extraction, we build a unit dictionary by collecting keywords from unit names. In addition, we need to build a negative word dictionary by collecting keywords from program names.

5.8.1 Unit Dictionary and Negative Word Dictionary

We build the unit dictionary the same way as we build the division dictionary. We collect a total of about 350 keywords for the unit dictionary. One important observation is that unit list and program list often appear on the same page and a unit

Algorithm 5.10: Division URL Retrieval

```
input  : DivisionNodes, PageNodes
output: DivisionNodes
for i=0 to DivisionNodes.size-2 do
    for j=DivisionNodes [i].position+1 to DivisionNodes [i+1].position-1 do
        if PageNodes [j] is a link node and its anchor text contains "http",
        "Home", "Homepage", "Website", "Visit", or "More" then
            DivisionNodes [i].url = PageNodes [j].url;
            break;
        end
    end
end
if DivisionNodes [0].url != null then
    for j=DivisionNodes [DivisionNodes.size-1].position+1 to PageNodes.size-1
    do
        if PageNodes [j] is a link node and its anchor text contains "http",
        "Home", "Homepage", "Website", "Visit", or "More" then
            DivisionNodes [DivisionNodes.size-1].url = PageNodes [j].url;
            break;
        end
    end
end
if DivisionNodes [0].url == null then
    for j=DivisionNodes [i].position+1 to DivisionNodes [i+1].position-1 do
        if PageNodes [j] is a link node then
            DivisionNodes [i].url = PageNodes [j].url;
            break;
        end
    end
    if DivisionNodes [0].url != null then
        for j=DivisionNodes [DivisionNodes.size-1].position+1 to
        PageNodes.size-1 do
            if PageNodes [j] is a link node then
                DivisionNodes [DivisionNodes.size-1].url = PageNodes [j].url;
                break;
            end
        end
    end
end
return DivisionNodes;
```

name and a program name can share the same keywords. To prevent program lists from being identified as unit lists, we build a negative word dictionary by collecting words that can only be present in a program list. Table 12 and 13 give two examples on how we build the negative word dictionary.

Table 12: Building Negative Word Dictionary for Unit List: One

Item in Program List	**Negative words collected**
Bachelor of Humanities and Arts Intercollege Degree Program	Bachelor, Degree, Program
Bachelor of Science and Arts Intercollege Degree Program	Bachelor, Degree, Program
Bachelor of Science in Computational Biology	Bachelor
Health Professions Program	Program
Science & Humanities Scholars Program	Program
Secondary Major in Human Computer Interaction	Major

Table 13: Building Negative Word Dictionary for Unit List: Two

Item in Program List	**Negative words collected**
Cognitive Science BA	BA
Computational Math BSc	BSc
Computer Science BA, BSc, iBA, iBSc	BA, BSc, iBA, iBSc
Digital Design Certificate	Certificate

We collect a total of about 50 negative words including various degree names and their corresponding acronyms, “Program”, “Institute”, “Center”, “Degree”.

5.8.2 Unit List Identification

The unit list identification algorithm is similar to that of division list. The detailed algorithm is given in Algorithm 5.11.

Algorithm 5.11: Identifying Unit List

input : A list of value pairs $(text_1, url_1), (text_2, url_2), ..., (text_n, url_n)$
output: A boolean value

```
1  counter = 0;
2  for i ← 1 to n do
3  |  if text_i has no more than 10 words and text_i contains no digits and at least
   |  one word from the unit dictionary and url_i does not contain any word from
   |  the negative word dictionary then
4  |  |  increment counter;
5  |  end
6  end
7  if counter >= n * 2/3 then
8  |  return true;
9  else
10 |  return false;
11 end
```

5.8.3 Unit List Priority

After applying the identification algorithm on unit list candidates, we often get more than one list as the result. It is true because program lists under each unit can be on the same page as the unit list and both kinds of lists can share some field keywords. Another case is that there can be two unit lists of different forms on the same page. We define three priority rules so that only the most likely one is returned. The three rules are listed based on their priorities.

Unit Name Keyword (Rule one) At least 2/3 list items contain keywords including "Department" and "School". Both keywords are treated as case insensitive.

List Heading (Rule two) The list heading contains keywords including "Departments" and "Schools". Both keywords are treated as case insensitive.

Candidate Generation Type (Rule three) The same as the division list.

In particular, lists satisfying all three rules are considered first. Then lists satisfying both rule one and rule three are considered. Lists satisfying both rule two and rule three are considered next. Finally, lists satisfying rule three are considered.

5.8.4 The Complete Algorithm

The complete algorithm works in three iterations. First we use the seven list generation routines to generate candidate lists from the unit list page. Second we use the identification algorithm to identify potential unit lists from the candidate lists. Last we apply the priority rules to all potential unit lists and return the most likely list. If no unit lists are identified in the second iteration, we return an empty list. The complete algorithm is given in Algorithm 5.12.

5.9 Faculty List Extraction

Given the faculty list page, we want to extract information of each faculty member from the list. In a faculty list page, information for faculty members is grouped together in a data region. In terms of DOM tree structure of the page, all information blocks for these faculty members are directly under a common DOM element (i.e., their parent element) and these blocks are in a contiguous region. We often call such an information block a data record. Another observation is that the information for each faculty member is encoded with the same or similar HTML tags. This is because the faculty list page is usually generated from a backend faculty database with a fixed template. Unfortunately, we do not have direct access to such a database. To extract

Algorithm 5.12: Unit List Extraction

input : A web page as given by its URL
output: A list of value pairs or an empty list

```
1  Render the page using CSSBox, retrieve text, url, x, y, height, style, index
   information for each text node on the page and store the information with a
   NodeInfo object. A list L of NodeInfo objects are ready for processing;
2  units ← ∅;
3  candidates ← NestedTiledCandidates(L) ∪ NestedCandidates(L) ∪
   TiledCandidates(L) ∪ VerticalCandidates(L) ∪
   ApproximateIndexedCandidates(L) ∪ StrictIndexedCandidates(L) ∪
   HorizontalCandidates(L);
4  foreach list l in candidates do
5  |  if IdentifyUnits(l) then
6  |  |  units ∪l;
7  |  end
8  end
9  first, second, third = null;
10 foreach list l in units do
11 |  if first == null and RuleOneTrue(l) and RuleTwoTrue(l) then
12 |  |  first = l;
13 |  else if second == null and RuleOneTrue(l) then
14 |  |  second = l;
15 |  else if third == null and RuleTwoTrue(l) then
16 |  |  third = l;
17 |  if first!= null and second!= null and third!= null then
18 |  |  break;
19 |  end
20 end
21 if first!= null then
22 |  return first;
23 else if second!= null then
24 |  return second;
25 else if third!= null then
26 |  return third;
27 else if units is not empty then
28 |  return the first list in units;
29 return ∅;
```

faculty information from the faculty list page, we adopt an approach based on pattern mining. In particular, we borrow the idea from a paper on data record mining [19]. The entire faculty list extraction algorithm is a three-step process. First of all, we try to generate candidate lists by mining data records on the page. Then, we identify the faculty list by incorporating a surname database. Finally, we extract different kinds of information for each faculty member from the list.

5.9.1 Candidate List Generation

We take advantage of the results on data record mining from [19] for the candidate list generation task. Given a web page that contains structured data records, the goal of their work is to segment these data records, extract data items/fields from them and put the data in a database table. Our goal is very similar to theirs except that they are working on general purpose data record mining tools while we are focusing on extracting data records that contain faculty member information. There is an implementation of the algorithm in pure Java by Sigit Dewanto [50], which we use as the basis of our candidate list generation algorithm. Since both the algorithm [19] and the implementation details [50] are available, we only briefly describe the idea, explain the input, output and parameters for the algorithm, and finally go through the algorithm skeleton.

The Idea of Mining Data Records

Data records on web pages are typically generated from a backend database with a fixed template and arranged in a contiguous region under a common parent node. We call such a contiguous region data region. Since data records are encoded based on a fixed template, they share the same or similar encoding tag sequences. The algorithm first tries to mine all data regions from the page, then it identifies individual data records within each data region and finally maps data items/fields in data records to

table columns. When deciding the similarity between two data records, they build tag trees from the HTML elements that contain the data records and then compare the tag trees using tree edit distance.

Input, Output and Parameters

The input of the algorithm is the faculty list page. The output is a list of database tables where each table row corresponds to a data record and each table column corresponds to an item/field of these data records. One important parameter is the similarity threshold between two data records. In our work, it always takes the value of 0.8, which is empirically set. Another important parameter is the tag node number of each data record. A data record is considered a generalized node, which can contains single or multiple HTML tag nodes. For example, the data record for each faculty member can consist of only one tag node or up to five tag nodes.

Algorithm Skeleton

The algorithm skeleton (without details) for mining data records is given in Algorithm 5.13.

Algorithm 5.13: Mining Data Records

input : Faculty list page URL, Parameter for tag nodes NumNodes
output: A list of database tables

```
1 Result = ∅;
2 DataRegion [] regions = MiningDataRegion (URL,NumNodes,0.8);
3 foreach DataRegion region in regions do
4 |   DataRecord [] records = IdentifyDataRecords (region);
5 |   Table table = AlignDataItems (records,0.8);
6 |   add table to Result;
7 end
8 return Result;
```

5.9.2 Faculty List Identification

Now that we obtain a table filled with structured data where each column contains a specific kind of data item. If the table contains information from a faculty list, there must be a column where every data item is a faculty member name. We determine if a table is a faculty list table by utilizing a surname database. In particular, we check if there exists one column in the table where each row item contains a surname from the database. We use surnames occurring 100 or more times from a census of year 2000 [51] by the United States Census Bureau. The identification algorithm is given in Algorithm 5.14.

Algorithm 5.14: Identifying Faculty Table

input : A table corresponding to a data region
output: A boolean value indicating whether the table contains faculty list information

```
1  foreach column col in the given table Table do
2  |  counter = 0;
3  |  foreach row in the given table Table do
4  |  |  if Table [row][col] contains one surname from the database then
5  |  |  |  increment counter;
6  |  |  end
7  |  end
8  |  if counter >= Table.rows * 2/3 then
9  |  |  return true;
10 |  end
11 end
12 return false;
```

5.9.3 The Complete Algorithm

We optimize the algorithm by going through up to 5 iterations. In the first iteration, we retrieve all data regions where each data record consists of one tag. If any faculty information tables are identified, we just return these tables. Otherwise, we go to

the second iteration and retrieve all data regions where each data record consists of two tags. If any faculty information tables are identified, we just return these tables. Otherwise, we proceed to further iterations. This decision is based on the observation that faculty information blocks are mostly encoded by one tag and second mostly encoded by two tags. There are cases where these blocks are encoded by more than two tags. We assume that information blocks are encoded by no more than five tags. The complete algorithm for converting a faculty list page to a list of faculty information tables is given in Algorithm 5.15.

Algorithm 5.15: Retrieving Faculty Information Tables

input : Faculty list page URL
output: A list of faculty information tables

```
1  Result = ∅;
2  for i ← 1 to 5 do
3  |  Table[] tables = MiningDataRecords (URL, i);
4  |  for j ← 0 to tables.size do
5  |  |  if IdentifyFacultyTable (tables[j]) then
6  |  |  |  add tables[j] to Result;
7  |  |  end
8  |  end
9  end
10 return Result;
```

5.9.4 Faculty Attribute Extraction

Now that we have not only identified all repeated information blocks (the data records) but also aligned all data fields (data items within each data record) within each block. If we regard the entire repeated structure as a table, each repeated information block can be regarded as a single row of the table and each aligned data field can be regarded as a single column of the table. As the experimental results show, there are three types of faculty information tables. In the first type, each row contains information for exact one faculty member, which is the most common case

for real-world faculty lists. One example of this type is given in Figure 43. In the second type, each row contains information for multiple faculty members where the information is sorted by individual faculty member. For example, column 1 up to column 4 contain information for the first faculty member while column 5 up to column 8 contain information for the second faculty. One example of this type is given in Figure 44. In the third type, each row contains information for multiple faculty members where the information is sorted by individual attribute. For example, column 1 up to column 3 contain photo information of three faculty members while column 4 up to 6 contain name information of the same three faculty members. One example of this type is given in Figure 45.

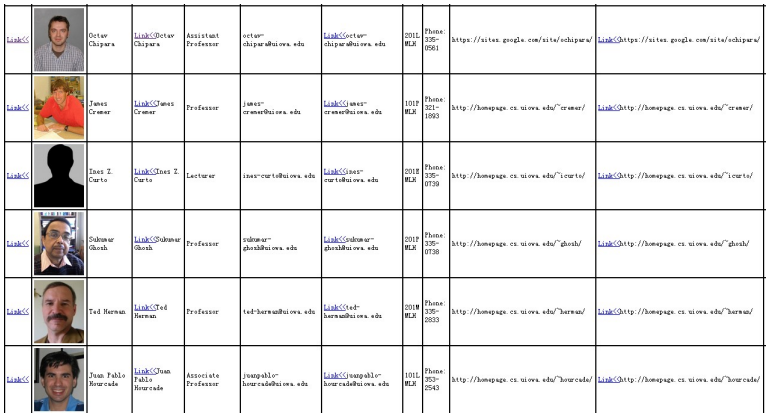


Figure 43: Type one table: each row contains information for exact one faculty member.

Figure 44: Type two table: each row contains information for multiple faculty members sorted by faculty member.

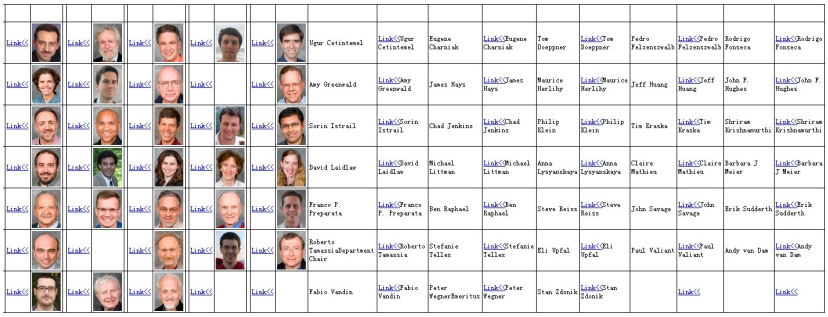


Figure 45: Type three table: each row contains information for multiple faculty members sorted by attribute.

Given these tables, the goal is to extract various attributes for each faculty member. In particular, we are interested in extracting the following attributes: photo, name, homepage, position, phone, fax, email. As seen from the table examples, faculty members do not necessarily have all these attributes present in the faculty list. Each faculty member can have as few as two attributes (i.e., name and homepage) and as many as over ten attributes (including other attributes such as office and research interests). The set of actual attributes in a particular faculty list varies greatly from department to department. To effectively extract attributes from these tables, we first check what type of table it is and then employ different algorithms for different types. The table type checking algorithm is given in Algorithm 5.16.

Algorithm 5.16: Faculty Information Table Type Checking

input : Faculty information table
output: Type of the table

```
1  nameColumnCount = 0;
2  tableColumnCount = total number of columns in the table;
3  nameColumns = ∅;
4  foreach column col in the table do
5  |  if IsNameColunm (col) then
6  |  |  increment nameColumnCount;
7  |  |  add column index of col to nameColumns;
8  |  end
9  end
10 if nameColumnCount == 1 then
11 |  return type one;
12 else
13 |  normalizedNameColumns = ∅;
14 |  indexDifference = nameColumns [0];
15 |  foreach column index index in nameColumns do
16 |  |  add index− indexDifference to normalizedNameColumns;
17 |  end
18 |  if normalizedNameColumns [1] - normalizedNameColumns [0] ==
   |  tableColumnCount- normalizedNameColumns.last then
19 |  |  return type two;
20 |  else
21 |  |  return type three;
22 |  end
23 end
```

The checking algorithm first counts the total number of name columns and records these columns in a list. If there is exactly one name column in the table, we decide that it is a type one table. Otherwise, we shift every column recorded in the list by the difference of the first name column and the first column of the table. We call the new list a normalized name column list in the algorithm. If the table is of type two, each column index in the new list must be the starting column for each faculty member. Thus, the difference between the last column index in the new list and the total number of table columns must be equal to the difference between adjacent column indices in the new list. If the equality holds, we decide that it is a type two table. Otherwise, it is a type three table.

Before we give the complete algorithm for faculty attribute extraction, we first define the name column checking method used in the table type checking algorithm as well as column checking methods for photo, homepage, position, phone, fax and email.

isNameColumn There are at least 2/3 row items where each item is not a link, not an email address, and contains one name from the surname database.

isPhotoColumn There are at least 2/3 row items where each item is an img element and its width/height ratio must be less than 3.

isHomepageColumn There are at least 2/3 row items where each item is a link element and its anchor text either contains one of the corresponding faculty name, “Web”, “Home”, “Site” and “Profile” or starts with “http”.

isPositionColumn There are at least 2/3 row items where each item contains keywords including “Professor”, “Faculty”, “Assistant” and “Associate”.

isPhoneColumn There are at least 2/3 row items where each item matches a predefined regular expression for phones.

isFaxColumn There are at least 2/3 row items where each item matches a predefined regular expression for phones.

isEmailColumn There are at least 2/3 row items where each item matches a predefined regular expression for emails.

In name column checking, we require that each item be neither a link nor an email address because items for homepage and email usually contain names from the surname database. In photo column checking, we require that the width/height ratio be less than 3 because emails are sometimes encoded as an image. In homepage column checking, we first check if the anchor text contains the corresponding faculty name because in most cases homepage is encoded as the underlying hyperlink of the name. In position column checking, we choose four most frequent keywords in real-world positions to identify the position column. In both phone and fax column checking, we first use the same regular expression to match against row items. Then we use three extra rules to distinguish between phone and fax. In particular, we check if the item itself contains the keywords "Phone" and "Fax" and if the corresponding item in the previous column contains the keywords "Phone" and "Fax". If both conditions fail, we identify the first matched item as phone and the second one as fax. In email column checking, we simply check the row item against a predefined regular expression.

We first give the algorithm for faculty attribute extraction from type one table in Algorithm 5.17.

For a type two table, we first divide the table into multiple type one tables by computing the column range for each faculty member in one row. For example, if there are four faculty members in each row of the table, we divide the table into four

Algorithm 5.17: Faculty Attribute Extraction: Type One Table

input : Faculty information table
output: List of Faculty objects with attributes initialized

```
1  Faculty[] faculties = new Faculty[table.rows];
2  foreach Column col in the table do
3  |  if isNameColumn (col) then
4  |  |  for i=0 to table.rows-1 do faculties[i].name = table[i][col];
5  |  else if isPhotoColumn (col) then
6  |  |  for i=0 to table.rows-1 do faculties[i].photo = table[i][col];
7  |  else if isHomepageColumn (col) then
8  |  |  for i=0 to table.rows-1 do faculties[i].homepage = table[i][col];
9  |  else if isPositionColumn (col) then
10 |  |  for i=0 to table.rows-1 do faculties[i].position = table[i][col];
11 |  else if isPhoneColumn (col) then
12 |  |  for i=0 to table.rows-1 do faculties[i].phone = table[i][col];
13 |  else if isFaxColumn (col) then
14 |  |  for i=0 to table.rows-1 do faculties[i].fax = table[i][col];
15 |  else if isEmailColumn (col) then
16 |  |  for i=0 to table.rows-1 do faculties[i].email = table[i][col];
17 |  end
18 end
19 return faculties;
```

type one tables. Then we extract faculty attributes from each of the type one tables and merge the results from all tables. The algorithm for type two table attribute extraction is given in Algorithm 5.18.

Algorithm 5.18: Faculty Attribute Extraction: Type Two Table

input : Faculty information table
output: List of Faculty objects with attributes initialized

```
1  nameColumns = ∅;
2  nameColumnCount = 0;
3  foreach column col in the table do
4  |  if IsNameColunm (col) then
5  |  |  add column index of col to nameColumns;
6  |  |  increment nameColumnCount;
7  |  end
8  end
9  Faculty[] faculties = new Faculty[table.rows * nameColumnCount ];
10 normalizedNameColumns = ∅;
11 indexDifference = nameColumns [0];
12 foreach column index index in nameColumns do
13 |  add index− indexDifference to normalizedNameColumns;
14 end
15 for i=0 to normalizedNameColumns.size-2 do
16 |  treat column normalizedNameColumns [i] through column
   |  normalizedNameColumns [i+1]-1 as a type one sub-table;
17 |  use the algorithm for type one table to extract the sub-table and add the
   |  result to faculties;
18 end
19 treat column normalizedNameColumns.last through column table.columns-1 as a
   type one sub-table;
20 use the algorithm for type one table to extract the sub-table and add the result
   to faculties;
21 return faculties;
```

In a type three table, each row contains information for multiple faculty members and the information is sorted by individual attribute rather than individual faculty member. One important observation is that the order of all attributes for each faculty member is consistent, which makes it possible to first identify all columns for each attribute and then map these columns to different faculty members by the order. The

algorithm for type two table attribute extraction is given in Algorithm 5.19.

5.10 Faculty Homepage Extraction

The goal is to extract photo, position, phone, fax, email information from a faculty member's homepage. We first identify various features for each of these faculty attributes. Based on these features, we design rules to extract them from the homepage.

5.10.1 Feature Identification

For extracting profile photo on faculty member's homepage, we derive six features and assign weight to each of them.

Table 14: Photo Feature Weight Table

Feature	**Weight**	**Description**
Face detection	60	We check if there is a human face in the current photo
Image height/width ratio	10	We set the ratio to be between 0.8 and 2
Image file name	10	We check if the file name contains (part of) the person's name
Positive keywords	10	We look for positive words such as "myself", "me", "portrait" in the image file name
Image attribute	10	We check if the "ALT" attribute contains (part of) the person's name
Negative keywords	-50	We look for negative words such as "logo", "email" in the image file name

For face detection and image height/width retrieval, we take advantage of the OpenCV library [52]. To extract a faculty member's position, we make use of a

Algorithm 5.19: Faculty Attribute Extraction: Type Three Table

input : Faculty information table
output: List of Faculty objects with attributes initialized

```
1  nameColumns = ∅;
2  photoColumns = ∅;
3  homepageColumns = ∅;
4  positionColumns = ∅;
5  phoneColumns = ∅;
6  faxColumns = ∅;
7  EmailColumns = ∅;
8  foreach column col in the table do
9      if IsNameColumn (col) then
10         add column index of col to nameColumns;
11     else if IsPhotoColumn (col) then
12         add column index of col to photoColumns;
13     else if IsHomepageColumn (col) then
14         add column index of col to homepageColumns;
15     else if IsPositionColumn (col) then
16         add column index of col to positionColumns;
17     else if IsPhoneColumn (col) then
18         add column index of col to phoneColumns;
19     else if IsFaxColumn (col) then
20         add column index of col to faxColumns;
21     else if IsEmailColumn (col) then
22         add column index of col to EmailColumns;
23     end
24 end
25 Faculty[] faculties = new Faculty[table.rows * nameColumns.size];
26 for i=0 to nameColumns.size-1 do
27     for j=0 to table.rows-1 do
28         Faculty faculty = new Faculty ();
29         faculty.name = table[j][NameColumns[i]];
30         if photoColumns.size == nameColumns.size then faculty.photo =
           table[j][PhotoColumns[i]];
31         if homepageColumns.size == nameColumns.size then faculty.homepage
           = table[j][HomepageColumns[i]];
32         if positionColumns.size == nameColumns.size then faculty.position =
           table[j][PositionColumns[i]];
33         if phoneColumns.size == nameColumns.size then faculty.phone =
           table[j][PhoneColumns[i]];
34         if faxColumns.size == nameColumns.size then faculty.fax =
           table[j][FaxColumns[i]];
35         if EmailColumns.size == nameColumns.size then faculty.email =
           table[j][EmailColumns[i]];
36     end
37 end
38 return faculties;
```

dictionary of common positions in the university domain. We first generate position candidates based on the dictionary and then identify the one closest to the faculty member's name as the position value. The dictionary is sorted by the number of words in each position. The longest position is always matched first. For example, we match "Associate Professor" before "Professor". To locate the name on homepage, we use the name extracted from the faculty list. For phone, fax, email extraction, we use regular expressions combined with possible prefixes such as "phone:", "fax:", "email:". Some emails are displayed using images in place of text. There can be other people's (e.g., assistant, admin) contact information on the same page as well.

5.10.2 The Algorithm

We use two types of regular expressions to identify phone and fax.

Format Regex Regular expression that matches the format of a phone or fax number, e.g.,

```
(\\d{3}[\\s\\.\\-]? (\\s)*\\d{4})
```

.

Keyword Regex Regular expression that not only matches the format but also the keywords in front, e.g.,

```
(Phone (\\s)*:? (\\s)*\\d{3}[\\s\\.\\-]? (\\s)*\\d{4})
```

.

The complete algorithm for homepage extraction is given in Algorithm 5.20.

Algorithm 5.20: Faculty Homepage Extraction

input : Faculty member's homepage given as a URL
output: Photo, position, phone, fax, email

```
1  Parse the page into a DOM object;
2  Retrieve all img elements from the object;
3  foreach image photoCandidate in retrieved img elements do
4  |  compute the total score for photoCandidate according to the photo feature
   |  weight table;
5  |  if the total score for photoCandidate is less than 60 then
6  |  |  discard photoCandidate;
7  |  end
8  end
9  return the candidate with the highest score as photo value;
10 if there is only one match by KeywordRegex (phone) then
11 |  return the match as phone value and remove the match from input;
12 else if there is more than one by KeywordRegex (phone) then
13 |  return the first match without negative words "assistant", "admin",
   |  "secretary" in front as phone value and remove all matches from input;
14 end
15 if there is only one match by KeywordRegex (fax) then
16 |  return the match as fax value and remove the match from input;
17 else if there is more than one by KeywordRegex (fax) then
18 |  return the first match without negative words "assistant", "admin",
   |  "secretary" in front as fax value and remove all matches from input;
19 end
20 if there is at least one match by FormatRegex (phone) and phone value is not
   returned yet then
21 |  return the first match as phone value;
22 else if phone value is already returned but fax value is not returned yet then
23 |  return the first match as fax value;
24 end
25 if there is only one match by KeywordRegex (email) then
26 |  return the match as email value;
27 else if there is more than one by KeywordRegex (email) then
28 |  return the first match without negative words "assistant", "admin",
   |  "secretary" in front as email value;
29 end
30 else if there is an img element preceded by "Email:" then
31 |  return the image as email value;
32 end
33 Retrieve position candidates by matching from position dictionary;
34 Identify faculty name on the page by matching name extracted from faculty
   list;
35 return the candidate closest to faculty name on the page as position value;
```

5.11 University General Information Extraction

Given the university name, the goal is to extract general information such as president, founding time, address, and motto from each university's Wikipedia page. Almost every university has a Wikipedia page, which contains its general information. The general information is consistently presented using a table on the right side of the page. Common information includes motto, founding time, type, endowment, president, provost, location, number of students and academic staff.

5.11.1 Wikipedia Page Analysis

By investigating the Wikipedia pages of 100 universities, we are able to make the following general observations:

Page URL The Wikipedia page for a given university always has a URL starting with `http://en.wikipedia.org/wiki/` followed by the university name with space replaced by underscore. For example, the URL for Stanford University is `http://en.wikipedia.org/wiki/Stanford_University`.

Table Location The table containing general information has at least one of the following attributes: Motto, Established, Location, Website.

Value Pair The attribute name and value are encoded with th and tr respectively on each row.

There are about 40 attributes found in the 100 universities. For the same attribute there can be two different labels, which we need to merge. For values of the same attribute in different universities there can be different value formats, which we need to normalize. Figure 46 shows the attribute table of Carleton University on its Wikipedia page.

Figure 46: Attribute table of Carleton University on Wikipedia

5.11.2 The Algorithm

The detailed algorithm for extracting general information from university's Wikipedia page is given in Algorithm 5.21.

5.12 Division List Page Candidate Retrieval and Extraction Result Integration

Given a university homepage, the goal is to first retrieve all division list page candidates. Then we apply the division list extraction algorithm on these candidates. Finally, we integrate results from different candidates using three priority rules. We have two routines for candidate page retrieval. One is based on navigation heuristics. The other one is based on link traversal. As an experienced university website user, we follow links containing certain keywords to find the division list page. These keywords

Algorithm 5.21: University General Information Extraction

input : University name
output: Motto, address, president, founding time

```
1  Generate the Wikipedia page URL as
   http://en.wikipedia.org/wiki/University_Name;
2  Parse the page into a DOM object and retrieve all table elements;
3  foreach table t in retrieved tables do
4  |  if t contains a row whose th text is "Motto", "Established", "Location" or
   |   "Website" then
5  |  |  foreach row r in t do
6  |  |  |  if r's th text is "Motto" then
7  |  |  |  |  return r's td text as motto value;
8  |  |  |  end
9  |  |  |  if r's th text is "Established" then
10 |  |  |  |  return r's td text as founding time value;
11 |  |  |  end
12 |  |  |  if r's th text is "Location" then
13 |  |  |  |  return r's td text as address value;
14 |  |  |  end
15 |  |  |  if r's th text is "President" then
16 |  |  |  |  return r's td text as president value;
17 |  |  |  end
18 |  |  end
19 |  end
20 end
```

are “Faculties”, “Schools”, “Colleges”, “Academics”, “Divisions”, “Academic Units”. The heuristics-based retrieval takes advantage of these keywords and is both effective and efficient. We do not get to the link traversal routine unless we are not able to extract division information from candidate pages retrieved by the heuristics-based routine.

5.12.1 Heuristics-based Retrieval

We first introduce some notations used in the algorithm. The notations and their corresponding meanings are given in Table 15. The algorithm for retrieving division list page candidates using heuristics is given in Algorithm 5.22.

Table 15: Navigation Notations

Notation	**Meaning**
Page⟨⟩	the input page
Page⟨Schools⟩	the landing pages by following links containing “Schools” from Page⟨⟩
Page⟨Academics, Schools⟩	the landing pages by following links containing “Academics” from Page⟨Academics⟩

5.12.2 Traversal-based Retrieval

Using breadth first traversal, we start from university homepage and retrieve all links of depth no more than 3. During traversal, we remember links that have been visited before so that they will not be visited again; and we ignore links that point to media type of resources (e.g., word document, spreadsheet document, audio and video files). The page retrieval algorithm based on link traversal is given in Algorithm 5.23.

Algorithm 5.22: Retrieving Division List Page Candidates Using Heuristics

input : University homepage
output: Division list page candidates

```
1  candidates = ∅;
2  candidates = candidates ∪ Page⟨⟩;
3  candidates = candidates ∪ Page⟨Faculties⟩;
4  candidates = candidates ∪ Page⟨Colleges⟩;
5  candidates = candidates ∪ Page⟨Divisions⟩;
6  candidates = candidates ∪ Page⟨Schools⟩;
7  candidates = candidates ∪ Page⟨Academics⟩;
8  candidates = candidates ∪ Page⟨Academics,Faculties⟩;
9  candidates = candidates ∪ Page⟨Academics,Colleges⟩;
10 candidates = candidates ∪ Page⟨Academics,Divisions⟩;
11 candidates = candidates ∪ Page⟨Academics,Schools⟩;
12 candidates = candidates ∪ Page⟨Academic Units⟩;
13 candidates = candidates ∪ Page⟨Academic Units,Faculties⟩;
14 candidates = candidates ∪ Page⟨Academic Units,Colleges⟩;
15 candidates = candidates ∪ Page⟨Academic Units,Divisions⟩;
16 candidates = candidates ∪ Page⟨Academic Units,Schools⟩;
17 return candidates;
```

Algorithm 5.23: Retrieving Division List Page Candidates Using Traversal

input : Starting page URL, Traversal parameter traversalDepth, Set of visited pages Visited, Already retrieved pages Candidates
output: Division list page candidates

```
1 candidates = ∅;
2 visited = ∅;
3 candidates = candidates ∪ URL;
4 links = Retrieve all but media type links from URL;
5 visited = visited ∪ links;
6 foreach link in links do
7 |   retrieveByTraversal (link, traversalDepth-1, visited, candidates);
8 end
9 return candidates;
```

5.12.3 Integration Rules

After applying the division extraction algorithm on candidate pages, we usually get more than one result from different pages. We define three priority rules to identify the most likely one. These three rules are listed in the order of priority.

Division Name Keyword (Rule one) At least 2/3 of the list items contain keywords including "Faculty", "College", "Division" and "School". All these keywords are treated as case insensitive.

List Heading (Rule two) The list heading contains keywords including "Faculties", "Colleges", "Divisions" and "Schools". All these keywords are treated as case insensitive.

Link Anchor (Rule three) The link anchor leading to the candidate page contains keywords including "Faculties", "Colleges", "Divisions" and "Schools". All these keywords are treated as case insensitive.

Lists satisfying all three rules are given the biggest priority. Lists satisfying two rules have priority over lists satisfying only one rule. If no lists satisfy any of the rules, the first list from the results is returned. The detailed algorithm is given in Algorithm 5.24. These three integration rules are also used to semantically define extracted divisions. In particular, the first matched keyword in the process will be used. If none of the three matchings succeeds, we just leave the divisions undefined.

5.12.4 The Complete Algorithm

Given a university homepage, we first retrieve all division list page candidates, then extract division information from these pages and finally integrate extraction results

Algorithm 5.24: Get Top Result Based on Priority Rules

input : Division list candidates
output: Division list

```
1  first, second, third, fourth, fifth, sixth, seventh = null;
2  foreach list l in candidates do
3  |  if first == null and ruleOneTrue(l) and ruleTwoTrue(l) and
   |  ruleThreeTrue(l) then
4  |  |  first = l;
5  |  else if second == null and ruleOneTrue(l) and ruleTwoTrue(l) then
6  |  |  second = l;
7  |  else if third == null and ruleOneTrue(l) and ruleThreeTrue(l) then
8  |  |  third = l;
9  |  else if fourth == null and ruleTwoTrue(l) and ruleThreeTrue(l) then
10 |  |  fourth = l;
11 |  else if fifth == null and ruleOneTrue(l) then
12 |  |  fifth = l;
13 |  else if sixth == null and ruleTwoTrue(l) then
14 |  |  sixth = l;
15 |  else if seventh == null and ruleThreeTrue(l) then
16 |  |  seventh = l;
17 |  if first!= null and second!= null and third!= null and fourth!= null and
   |  fifth!= null and sixth!= null and seventh!= null then
18 |  |  break;
19 |  end
20 end
21 if first!= null then
22 |  return first;
23 else if second!= null then
24 |  return second;
25 else if third!= null then
26 |  return third;
27 else if fourth!= null then
28 |  return fourth;
29 else if fifth!= null then
30 |  return fifth;
31 else if sixth!= null then
32 |  return sixth;
33 else if seventh!= null then
34 |  return seventh;
35 else if candidates is not empty then
36 |  return the first list in candidates;
37 return an empty list;
```

from different pages. We combine the three steps in one single algorithm as shown in Algorithm 5.25.

Algorithm 5.25: Division List Extraction Algorithm

input : University homepage URL
output: Division list

```
1  divisions = ∅;
2  candidates = retrieveByHeuristics (URL);
3  foreach page page in candidates do
4  |  divisions = divisions ∪ extractDivisions (Page);
5  end
6  if divisions is not empty then
7  |  return getTopResult (divisions);
8  end
9  candidates = retrieveByTraversal (URL);
10 foreach page page in candidates do
11 |  divisions = divisions ∪ extractDivisions (Page);
12 end
13 if divisions is not empty then
14 |  return getTopResult (divisions);
15 end
16 return an empty list;
```

5.13 Unit List Page Candidate Retrieval and Extraction Result Integration

Given division homepage, the goal is to first retrieve all unit list page candidates. Then we apply the unit extraction algorithm on these candidates. Finally, we integrate results from different candidates using three priority rules. Just like division list page candidate retrieval, we have two routines for unit list page candidate retrieval. One is based on navigation heuristics. The other one is based on link traversal. As an experienced division website user, we follow links containing certain keywords to find the unit list page. These keywords include “Departments”, “Academics”, “Academic

Units", "Programs" and "Schools". The heuristics-based retrieval takes advantage of these keywords and is both effective and efficient. We do not get to the link traversal routine unless we are not able to extract unit information from candidate pages retrieved by the heuristics-based routine.

5.13.1 Heuristics-based Retrieval

We use the same notations as in Table 15. The algorithm for retrieving unit list page candidates using heuristics is given in Algorithm 5.26.

Algorithm 5.26: Retrieving Unit List Page Candidates Using Heuristics

input : Division homepage
output: Unit list page candidates

```
1  candidates = ∅;
2  candidates = candidates ∪ Page⟨⟩;
3  candidates = candidates ∪ Page⟨Departments⟩;
4  candidates = candidates ∪ Page⟨Academics,Departments⟩;
5  candidates = candidates ∪ Page⟨Academic Units,Departments⟩;
6  candidates = candidates ∪ Page⟨Academics⟩;
7  candidates = candidates ∪ Page⟨Programs⟩;
8  candidates = candidates ∪ Page⟨Schools⟩;
9  candidates = candidates ∪ Page⟨Academics,Programs⟩;
10 candidates = candidates ∪ Page⟨Academic Units,Programs⟩;
11 candidates = candidates ∪ Page⟨Academics,Schools⟩;
12 candidates = candidates ∪ Page⟨Academic Units,Schools⟩;
13 return candidates;
```

5.13.2 Integration Rules

After applying the unit extraction algorithm on candidate pages, we usually get more than one result from different pages. We define three priority rules to identify the most likely one. These three rules are listed in the order of priority.

Unit Name Keyword (Rule one) At least 2/3 of the list items contain keywords

including “Department” and “School”. Both keywords are treated as case insensitive.

List Heading (Rule two) The list heading contains keywords including “Departments” and “Schools”. Both keywords are treated as case insensitive.

Link Anchor (Rule three) The link anchor leading to the candidate page contains keywords including “Departments” and “Schools”. Both keywords are treated as case insensitive.

Lists satisfying all three rules are given the biggest priority. Lists satisfying two rules have priority over lists satisfying only one rule. If no lists satisfy any of the rules, the first list from the results is returned. These three rules are also used to semantically define units.

5.13.3 The Complete Algorithm

Given division homepage, we first retrieve all unit list page candidates, then extract unit information from these pages and finally integrate extraction results from different pages. Note that besides the heuristics-based algorithm, we use the same traversal-based algorithm during page retrieval. We combine the three steps in one single algorithm as shown in Algorithm 5.27.

However, we are not done yet. We need to further verify that the returned list is indeed a unit list rather than a program list or something else. This is because some divisions may not contain units at all but only contain programs. We can decide whether it is a real unit list after we do a faculty list extraction under each “unit”. In particular, if we are able to extract faculty lists from at least half of the “units” in the list, we decide that it is a real unit list.

Algorithm 5.27: Unit List Extraction Algorithm

input : Division homepage URL
output: Unit list

```
1  units = ∅;
2  candidates = retrieveByHeuristics (URL);
3  foreach page page in candidates do
4  |  units = units ∪ extractUnits (Page);
5  end
6  if units is not empty then
7  |  return getTopResult (units);
8  end
9  candidates = retrieveByTraversal (URL);
10 foreach page page in candidates do
11 |  units = units ∪ extractUnits (Page);
12 end
13 if units is not empty then
14 |  return getTopResult (units);
15 end
16 return an empty list;
```

5.14 Faculty List Page Retrieval and Faculty Member Information Integration

Given a unit homepage, the goal is to retrieve the faculty list pages. There is usually one single page containing all faculty members. The only exception is that when there are too many faculty members in a unit, they can be divided into multiple pages. These pages are normally indexed alphabetically or numerically. In most cases we only need to retrieve one page for faculty list extraction while in the rest of cases we need to retrieve all indexed pages for faculty list extraction. Like division list page and unit list page retrieval, we use both a heuristics-based method and a traversal-based method to retrieve faculty list page candidates. Then, we propose an algorithm to identify the faculty list page or one of the faculty list pages (in case faculty members are divided into multiple pages). The identification algorithm

returns true if it recognizes the input page as a faculty list page and false otherwise. We need this extra identification algorithm because 1) the faculty list extraction algorithm tends to return results which are irrelevant when it processes a non-faculty list page (before it reaches the faculty list page while processing all page candidates) and 2) the faculty list extraction algorithm is less efficient in terms of both running time and computing power.

5.14.1 Faculty List Page Identification

Given a web page, the goal is to determine if it is a faculty list page. The algorithm is based on three observations:

The Invariant No matter how few or how many attributes are present in the faculty list, the faculty name is always there.

Vertical Alignment Each faculty name in most faculty lists is aligned vertically (i.e., has the same x-coordinate). In the case of tiled lists, names in each column are aligned vertically.

Visual Cues The names are easily distinguishable from other information in the list because they usually have different visual appearance in terms of font, color, length and height.

The identification algorithm works by detecting name lists from the input page. If it is able to detect a name list, it identifies the input page as a faculty list page. We assume that there are at least three faculty members in the list. The detailed algorithm is given in Algorithm 5.28.

Algorithm 5.28: Faculty List Page Identification

input : A web page under unit webiste
output: A boolean value, indicating if the web page is faculty list page

```
1  Render the page and retrieve text, x, height, style information of each text
   node to get a list of NodeInfo objects;
2  Cluster NodeInfo objects according to x, height, style to get a clustering where
   each element is a list of NodeInfo objects;
3  foreach element L in the clustering do
4  |  remove NodeInfo objects whose text does not match a name regular
   |  expression;
5  |  if L contains 3 or more NodeInfo objects then
6  |  |  counter = 0;
7  |  |  foreach NodeInfo object node in L do
8  |  |  |  if node's text contains a surname from the surname database then
9  |  |  |  |  increment counter;
10 |  |  |  end
11 |  |  end
12 |  |  if counter >= Size(L) * 3/5 then
13 |  |  |  return true;
14 |  |  end
15 |  end
16 end
17 return false;
```

5.14.2 Faculty List Page Selection Rules

We assume that there is only one faculty list page for each unit. In case the faculty list is divided into multiple pages, we only need to identify the first page and retrieve remaining pages separately. Using the faculty list page identification algorithm, we are able to identify more than one page as a faculty list page. For example, a staff page can be identified as well. Thus, we define three selection rules to further identify the real and only faculty list page.

Name List Heading (Rule one) If the name list extracted in the identification algorithm has a heading named "Faculty", we return the candidate page as the only faculty list page.

Link Anchor Path (Rule two) If the last link anchor in the path equals "Faculty", we return the candidate page as the only faculty list page.

Order of Candidates (Rule three) If both rule one and two do not work, we return the first candidate in the list as the only faculty list page.

5.14.3 Retrieving Faculty List Page Candidates by Heuristics

We first retrieve all links that contain "Faculty", "People", "Profile", "Directory" and "Staff" from the unit homepage and visit these links one by one. When we visit the links, we identify all links that contain "Faculty" on the landing pages. If a link on the landing page is not among the links retrieved from the homepage, we use the identification algorithm to check if it is a faculty list page. If none of the links on the landing page are identified as faculty list page, we go back and check the retrieved links from the homepage. The order in which we check all these second level and first level links is important. We always check the links containing "Faculty" first. For all links that contain "Faculty", we visit them in the order in which they appear on the

page.

5.14.4 The Complete Algorithm for Faculty List Page Retrieval

As in division list page retrieval and unit list page retrieval, we first retrieve the candidate pages by heuristics. If no result is returned using heuristics, we retrieve the candidate pages by link traversal. The traversal-based retrieval algorithm is the same as the one used for school and department. The complete algorithm for faculty list page retrieval is given in Algorithm 5.30.

5.14.5 Faculty List across Multiple Pages

For units which have too many faculty members, they often divide the faculty list into several sub-lists and place each sub-list on a separate page. These sub-lists are either indexed alphabetically or numerically. See Figure 47, 48, 49, 50 for examples. Using the faculty list page identification algorithm, we are able to obtain one of the sub-list pages (normally the first sub-list page in terms of indexing). Given the first sub-list page, the goal is to retrieve the rest of sub-list pages. First of all, we need to check if the given page is linked to any sub-list pages. Our algorithm is based on two assumptions. One assumption is that sub-list pages are indexed by natural numbers or capital English letters. The other assumption is that these indices are horizontally aligned on the given page. Both assumptions are true as far as our investigation goes. The algorithm for checking linked sub-list pages is given in Algorithm 5.31.

Algorithm 5.29: Retrieving Faculty List Page by Heuristics

input : The unit homepage
output: The faculty list page or null

```
1  candidates = ∅;
2  LevelOneUrls = ∅;
3  LevelTwoUrls = ∅;
4  Retrieve all links on the input page;
5  foreach link l in retrieved links do
6  |  if l's anchor text contains "Faculty", "People", "Profile", "Directory",
   |  "Staff" then
7  |  |  add l to LevelOneUrls;
8  |  end
9  end
10 foreach link l in LevelOneUrls do
11 |  retrieve all links on the landing page of l;
12 |  foreach link l in retrieved links do
13 |  |  if l's anchor text contains "Faculty" then
14 |  |  |  add l to LevelTwoUrls;
15 |  |  end
16 |  end
17 end
18 Remove all links in LevelTwoUrls whose URLs are present in LevelOneUrls;
19 foreach link l in LevelTwoUrls do
20 |  if IdentifyFacultyList (l) then
21 |  |  candidates ∪l;
22 |  end
23 end
24 foreach link l in LevelOneUrls do
25 |  if IdentifyFacultyList (l) then
26 |  |  candidates ∪l;
27 |  end
28 end
29 return candidates;
```

Algorithm 5.30: Faculty List Page Retrieval Algorithm

input : Unit homepage URL
output: Faculty List Page

```
1  candidates = retrieveByHeuristics (URL);
2  first, second, third = null;
3  foreach page page in candidates do
4  |  if first == null and RuleOneTrue(page) and RuleTwoTrue(page) then
5  |  |  first = page;
6  |  else if second == null and RuleOneTrue(page) then
7  |  |  second = page;
8  |  else if third == null and RuleTwoTrue(page) then
9  |  |  third = page;
10 |  if first!= null and second!= null and third!= null then
11 |  |  break;
12 |  end
13 end
14 if first!= null then
15 |  return first;
16 else if second!= null then
17 |  return second;
18 else if third!= null then
19 |  return third;
20 else if candidates is not empty then
21 |  return the first page in candidates;
22 candidates = retrieveByTraversal (URL);
23 first, second, third = null;
24 foreach page page in candidates do
25 |  if first == null and RuleOneTrue(page) and RuleTwoTrue(page) then
26 |  |  first = page;
27 |  else if second == null and RuleOneTrue(page) then
28 |  |  second = page;
29 |  else if third == null and RuleTwoTrue(page) then
30 |  |  third = page;
31 |  if first!= null and second!= null and third!= null then
32 |  |  break;
33 |  end
34 end
35 if first!= null then
36 |  return first;
37 else if second!= null then
38 |  return second;
39 else if third!= null then
40 |  return third;
41 else if candidates is not empty then
42 |  return the first page in candidates;
```

James Buehler, MD	Health Management and Policy	215-571-4015	james.buehler@drexel.edu
Igor Burstyn, PhD	Environmental and Occupational Health	(215) 762-2267	igor.burstyn@drexel.edu
Carla Campbell, MD, MS	Environmental and Occupational Health	(215) 762-4379	ccc57@drexel.edu

Showing 1-10 of 57
1 2 3 4 … 6 Next

Figure 47: Multiple-page faculty list example one

Figure 48: Multiple-page faculty list example two

Figure 49: Multiple-page faculty list example three

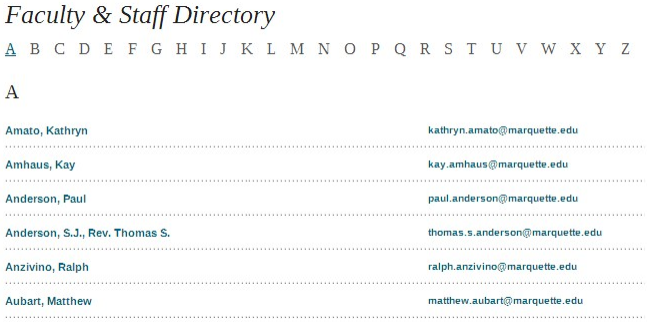


Figure 50: Multiple-page faculty list example four

If the algorithm identifies the given page as linked to sub-list pages, we first store

Algorithm 5.31: Checking Linked Sub-list Pages

input : A web page as given by its URL
output: A boolean value indicating if the page is linked to any sub-list pages

```
1  Render the page using CSSBox, retrieve text, url, y, height, style information
   for each text node on the page and store the information with a NodeInfo
   object. A list L of NodeInfo objects are ready for processing;
2  candidates ← HorizontalCandidates(L);
3  foreach list l in candidates do
4  |  numberCount = 0;
5  |  foreach NodeInfo node in l do
6  |  |  if node.text contains any natural number from 0 to 20 then
7  |  |  |  increment numberCount;
8  |  |  end
9  |  end
10 |  if numberCount *2 >= l.size then
11 |  |  return true;
12 |  end
13 |  letterCount = 0;
14 |  foreach NodeInfo node in l do
15 |  |  if node.text contains any capital letter from A to Z then
16 |  |  |  increment letterCount;
17 |  |  end
18 |  end
19 |  if letterCount *2 >= l.size then
20 |  |  return true;
21 |  end
22 end
23 return false;
```

the identified horizontal list of indices and then go through up to three iterations to retrieve all sub-list pages.

1. If the identified list contains a node whose text equals “ALL”, then we return the url of the node as the only faculty list page. In this case, there exists a single page containing all faculty members while there are sub-list pages containing faculty members sorted by letters.

2. If the identified list contains a node whose text equals “Next” and url is available, then we collect the url as one of the sub-list pages. We keep following the link of “Next” and collecting the corresponding url until the link “Next” is no longer clickable on the landing page.

3. If both rule one and two do not work, we return the first candidate in the list as the only faculty list page.

5.15 The Big Picture

In this section, we try to put everything together to complete the big picture for our extraction framework.

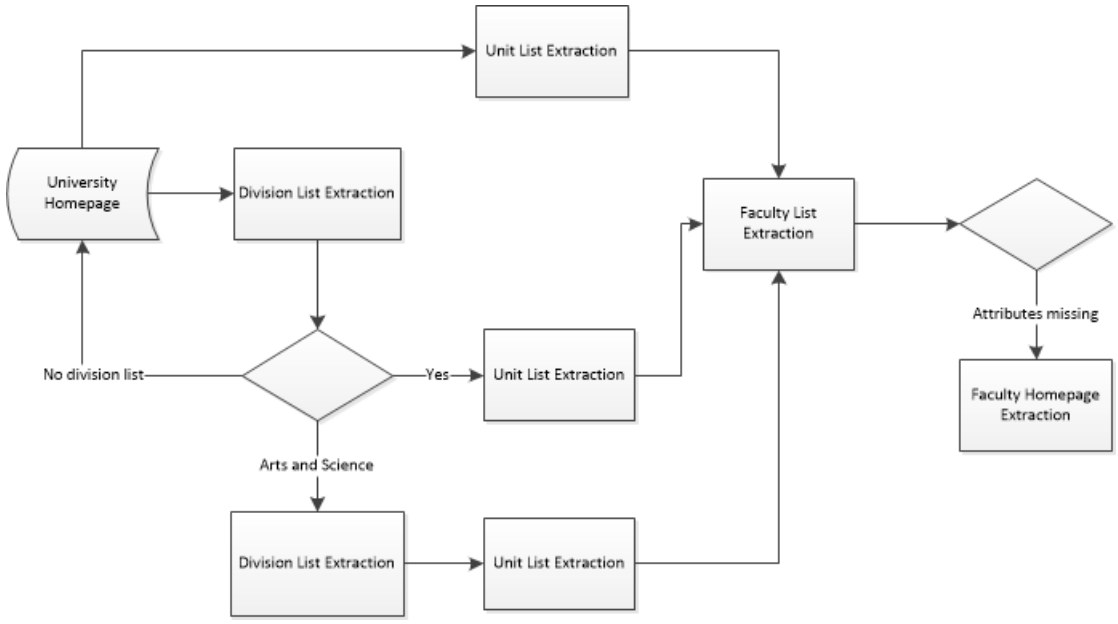


Figure 51: This shows the overall work flow of our extraction framework.

In Figure 51, we can see that the information source is a university homepage. University general information extraction from Wikipedia is an independent task and thus not included in the picture. Division list extraction, unit list extraction and faculty list extraction all consist of three stages: 1) retrieve candidate pages 2) extract information from candidate pages 3) integrate information extracted from different pages. Starting from the university homepage, we first do a division list extraction, which will result in three cases. The first case is that no division list is returned. In this case, we go back to university homepage and do a unit list extraction. The second case is that one division list is returned and one of the divisions in the list is named Arts and Science. In this case, we do a division list extraction for Arts and Science followed by a unit list extraction; for the rest of schools in the list, we just do a unit list extraction. Now every aforementioned path is arriving at a unit homepage. Then we do a faculty list extraction for all unit homepages. If all desired faculty information (including name, photo, homepage, position, phone, fax, email) is available from the faculty list, we are done; otherwise, we try to extract unavailable information from

the corresponding faculty homepages. Note that we only try to extract three levels down the academic unit hierarchy when the first-level school name is School of Arts and Science, College of Arts and Science, or Faculty of Arts and Science. This is an assumption we make in the current implementation. Also note that there are some divisions that are not further divided into units. In that case, we do faculty list extraction directly from the division website.

5.16 Experimental Results

In this section we summarize the experiments we conduct and explain the corresponding results. We evaluate our system by two measures—recall and precision. Recall is a measure of completeness, precision of correctness. To ensure we are able to extract quality data, we decide that our extraction system be precision-oriented. In particular, we only try to improve the recall provided that the precision is good enough (i.e., 80%).

5.16.1 Division List Extraction

We construct our algorithm based on 100 university websites. Then we test our algorithm against another 200 university websites. Results are summarized in Table 16.

Table 16: Division List Extraction Results

	Retrieved	**Correct**	**Precision**	**Recall**
Construction group	99	91	91.9%	91.0%
Testing group	194	156	80.4%	78.0%

In the construction group, there is only one division list which is not retrieved. It is due to the fact that there are too many irrelevant items in the division list so it

cannot pass the 2/3 threshold. Among the 8 incorrectly retrieved results, there are three cases where only partial results are retrieved. It is due to the fact that they put colleges, divisions and schools in different lists while our algorithm is only able to identify the most likely one. There is one case where all divisions are embedded in paragraphs rather than in a list and another case where only every other item in the list is vertically aligned, both of which our algorithm is not able to handle. There are two cases where the university does not have a division list but only has a big department list. Our algorithm identifies part of the department list as the result incorrectly. The last case is that the division list is a dropdown list and incorrectly merged with an irrelevant dropdown list since they are horizontally aligned. For the testing group, result analysis is given as follows. For the six cases where no results are retrieved, there are four reasons:

1. Too many irrelevant items are mixed with divisions in the same list, so it cannot pass the 2/3 threshold.

2. The division list is not available in the page source code since it might be generated by some script. The CSSBox library is not able to deal with such cases.

3. Items in the same list have different text decoration styles, as parsed by CSSBox. Thus these items cannot be grouped in one list.

4. The university website does not have a division list. There are two cases: one case is that the university is not divided into divisions at all while the other case is that there is not a division list explicitly available on the university website.

For the 38 cases where incorrect results are retrieved, there are four reasons:

1. The university website does not have a division list. An irrelevant list such as a program list, major list, minor list and degree list is retrieved incorrectly.

2. Colleges and schools are put in two different lists. Our algorithm is only able to retrieve the most likely one, so only partial results are retrieved.

3. Division lists that can be captured by the seven visual lists are only partially retrieved or cannot be retrieved. Three cases are found: one case is that divisions are embedded in paragraphs, another one is that divisions are aligned "centered" rather than "flush left", and the last case is that divisions are not aligned at all.

4. Division URLs are not retrieved or the retrieved URLs are incorrect.

First of all, the 100 universities for algorithm construction are all top universities in Canada and the US, so they are likely to cover most division keywords. The testing results show that the division keyword dictionary is fairly comprehensive. The 200 universities for algorithm testing are randomly selected from American universities, so that the testing results are likely to predict the applicability of our algorithm to general American universities. The results for the testing group degrade significantly compared to those for the construction group. There are two major reasons: 1) some small universities are not divided into divisions and only have a program list, major/minor list, or degree list available on their websites. 2) there are some irregular lists that cannot be captured by our seven visual list.

5.16.2 Unit List Extraction

We construct our algorithm based on 200 division websites from different universities. Then we test our algorithm against another 200 division websites. Results are summarized in Table 17.

Table 17: Unit List Extraction Results

	Relevant	**Retrieved**	**Correct**	**Precision**	**Recall**
Construction group	133	123	111	90.2%	83.5%
Testing group	148	138	118	85.5%	79.7%

In the construction group, there are 67 divisions which are not further divided into units. The 67 divisions are business schools, law schools, nursing schools, schools of medicine and schools of education. These divisions usually have a program list, which our algorithm might be able to extract by mistake. The post-extraction check for a faculty list under each extracted "unit" helps ensure that no program lists are retrieved by mistake. There are 10 cases where no results are retrieved. Three reasons are found: 1) the units cannot be rendered or properly rendered by the CSSBox library 2) the units are mixed with other irrelevant items so the list does not pass the 2/3 threshold 3) there are only two units in the division and they are embedded in paragraphs. There are 12 cases where incorrect results are retrieved. Four scenarios are found:

1. The division separates its school list from its department list. In other words, the schools and departments are not in a single unit list. In this scenario, only partial results can be retrieved.

2. The unit list cannot be captured by the seven visual lists. One case is that the unit list is made up of more than one horizontal list. Another case is that the unit items are in two columns but these two columns are not horizontally aligned to each other.

3. The unit URL cannot be retrieved. Since our URL retrieval algorithm works by checking the links following the unit name, it is not able to handle the case where the corresponding URL comes before the unit name. One example is that the URL is encoded by an image just above the unit name.

4. The first list item cannot be retrieved since it has different text decoration styles as parsed by the CSSBox library.

In the testing group, we can see that both precision and recall decline to some extent. The following four reasons might account for the decline:

1. The unit keyword dictionary is not comprehensive enough. Some keywords related to education and pharmacy are missing from the dictionary. As a result, three unit lists from education and pharmacy divisions cannot pass the 2/3 threshold.

2. Some unit lists cannot be captured by the seven visual lists. In particular, one unit list has all items aligned "flush right". Since these items do not share the same x-coordinate, they cannot be grouped in a list.

3. Units, programs and centres are mixed in a single big list. In such a case, we are either extracting the entire list or unable to extract the list.

4. The division is further divided into divisions, however, these divisions only function as a classification of the units but do not have their own URLs. In such a case, divisions and units are mixed in a nested list so we are able to extract the divisions. Since the division URLs are missing, our current algorithm is not able to further extract the units under each division.

5.16.3 Faculty List Extraction

We construct our algorithm based on 150 unit websites. Then we test our algorithm against another 150 unit websites. The results are summarized in Table 18.

Table 18: Faculty List Extraction Results

	Retrieved	**Correct**	**Precision**	**Recall**
Construction group	133	111	83.5%	74.0%
Testing group	134	114	85.1%	76.0%

We get similar results for both groups in terms of precision and recall. The following eight reasons account for those failure cases in both groups. The first four are reasons for not retrieved cases while the remaining four are reasons for incorrectly retrieved cases.

1. Faculty member names do not share the same x-coordinate. In such cases, the profile photos share the same x-coordinate and the x-coordinate of each name depends on the width of its corresponding photo. Since photos in some faculty lists are not resized to the same width, the names can have different x-coordinates.

2. The faculty list is generated by Javascript. Our current libraries are not able to handle Javascript.

3. No data records (repeated structures) are found. There are two cases: 1) the faculty members are not encoded as data records on the page 2) the similarity between two data records does not pass the 0.6 threshold.

4. The faculty list page cannot be identified. We identify the faculty list page by identifying a vertical name list on the page. In some cases, so many surnames in the faculty list are missing from the surname database that the vertical name list cannot pass the 3/5 threshold.

5. The faculty name is partially extracted because the last name and first name are split in two columns but our algorithm is unable to merge them.

6. There are either one or more faculty members missing from the results. We find two scenarios accounting for these cases. One scenario is that if one faculty member has fewer attributes than others, he or she can be missing from the faculty list. The other scenario is that several faculty members are contained in one single data record that comes last in the data region and has fewer faculty members than other data records. In both scenarios, the data records that have less information than others can be excluded from the data regions because they cannot pass the similarity threshold.

7. Irrelevant lists are retrieved along with the faculty list. Some navigation link lists are retrieved by mistake when the links contain too many keywords from the surname dictionary.

8. Attribute columns cannot be correctly identified. For example, we might identify the email column as name column when the domain part of the email is missing from the column and only available from outside the faculty list. Another example is that the regular expressions for phone and email fail to capture some unforeseen cases.

5.16.4 Overall Analysis

Table 19 is a summary of the above results as represented by F1-score. From Figure 52, we can see that there is a huge decline from the construction group to the testing group for division list extraction. This decline is mainly due to the way we select universities for both groups. We construct the algorithm based on top 100 universities while we test the algorithm against another 200 universities randomly selected from American universities. The irregularities of division lists from some small (and unforeseen) universities are mainly responsible for the decline. We choose the testing group so that it is able to test the applicability of our algorithm to arbitrary

American universities. We can also see that the decline for unit list extraction is not as significant as that for division list extraction. This is because there are fewer irregularities at unit list level than at division list level. Besides the list irregularities, the lack of some unit keywords accounts for the decline as well, which is not an issue at division list level. Finally, we have very similar results for both groups of faculty list extraction. In other words, we find few issues in the testing group which are not in the construction group.

Table 19: Summary of Overall Results

	Construction F1-score	**Testing F1-score**	**Decline**
Division list	91.4%	79.4%	12.0%
Unit list	86.7%	82.5%	4.2%
Faculty list	78.5%	80.3%	-1.8%

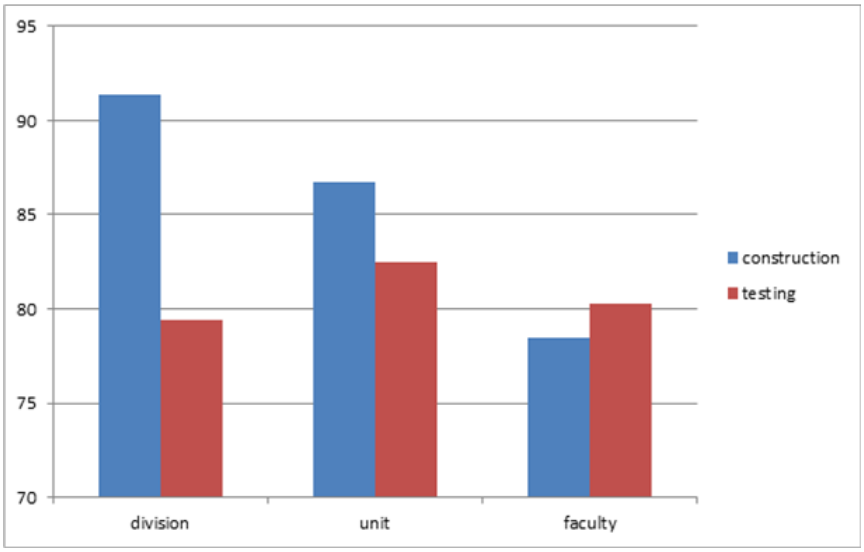


Figure 52: A contrast of construction group and testing group in terms of F1-score

5.16.5 Faculty Homepage Extraction

We construct our algorithm based on the same dataset as [49], in which they adopt a corpus-based approach by annotating examples and training a domain model using conditional random fields (CRF). There are 898 faculty homepages to be tested. We compare our result with their corpus-based result in Table 20 and Figure 53.

Table 20: Results Using Ruled-based Algorithm

Attribute	**Precision**	**Recall**	**F1-Score**	**F1-Score (corpus-based)**
Photo	94.3%	94.3%	94.3%	89.1%
Position	72.5%	70%	71.2%	69.4%
Phone	85.5%	80.7%	83.0%	91.1%
Fax	94.7%	89.5%	92.0	90.8%
Email	79.8%	85.7%	82.6%	80.4%

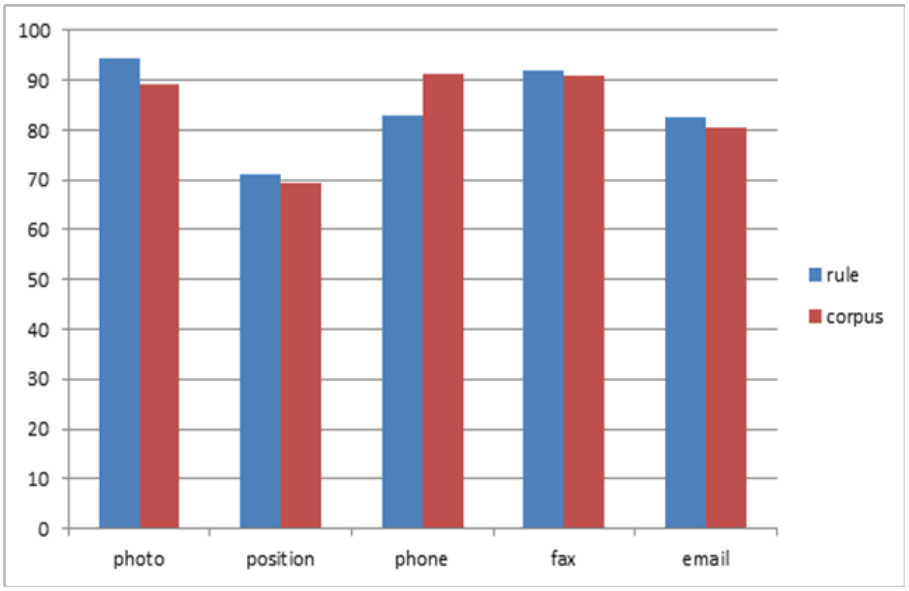


Figure 53: A contrast of rule-based approach and corpus-based approach in terms of F1-score

Our results are comparable to the results produced by the corpus-based approach. The result for position is not good enough because there are many uncommon position names not included in our dictionary. For example, positions such as Writer-in-Residence and CAS Member cannot be identified. Also, the current algorithm is not able to handle the case where position information is embedded in a natural language text paragraph (e.g., introductory paragraph). The result for phone is not as good as other results because there is often departmental contact information on faculty member introductory pages. Another reason is that without the prefix the regular expression can sometimes capture sequence of numbers that are similar to a phone number. The result for photo mostly depends on the outcome of face detection and is almost the best result we can obtain using the OpenCV library. Results for phone, fax, email are determined by the regular expressions and the prefix keywords and are almost the best we can obtain. There are two cases where it is almost impossible to extract the contact information. One case is that the information is displayed using an image, which our regular expressions are not able to handle. The other case is that the contact information is embedded in text that needs a little intelligence to interpret. Here are three examples: "To reach the above numbers from outside CMU, first dial 1-412-26", "You can reach me at firstname.lastname@cs.cmu.edu" and "Send me email at cs.cmu.edu, my user name is my three initials". We can see that fax has better results than phone. That is because when fax information is present in the page, it is usually prefixed by related keywords.

5.16.6 University General Information Extraction

We conduct an experiment on 100 universities and achieve perfect results. As the results show, we can easily extract university general information from its Wikipedia page as it is. Since Wikipedia pages are mainly created and maintained by volunteers, we put some effort to verify the data reliability by checking extracted information

against information found on official university websites. First of all, all 100 universities under investigation have a unique Wikipedia page in English. Second, the three attributes motto, founding time, location all have the correct information. These kinds of information barely change over time, which makes them really reliable to use. Finally, for president information, we find that one university has just had a new president, which has not been updated on Wikipedia yet. Based on our verification, we believe that the desired four pieces of information have very reliable presence on Wikipedia. However, this approach based on screen scraping is not robust against changes. As we change the underlying HTML and add more and more Javascript, the heuristics will degrade and eventually fail to work.

5.16.7 Performance Evaluation

Table 21 shows the environment we set up for our experiments. Since our program makes heavy use of network IO, we enable more threads than processors at the same time. During the tuning process, we decide the number of threads to be three times that of processors. We choose 64bit Java rather than 32 bit Java because 64bit Java allows unlimited memory usage (i.e., only restricted by the physical memory available) while 32bit Java does not allow the memory usage to be higher than 2G.

Table 21: Experimental Setup

Server info	32 processors, 32G memory, SunOS
Java version	Java 7, 64bit
Extraction scheme	One thread per university
Maximum number of threads	96

Figure 54 illustrates the running times regarding the numbers of universities being

extracted at the same time. The vertical axis denotes the running time in terms of days while the horizontal axis denotes the number of universities being extracted at the same time. From the figure, we can see that it demands significantly more time when the number of universities increases.

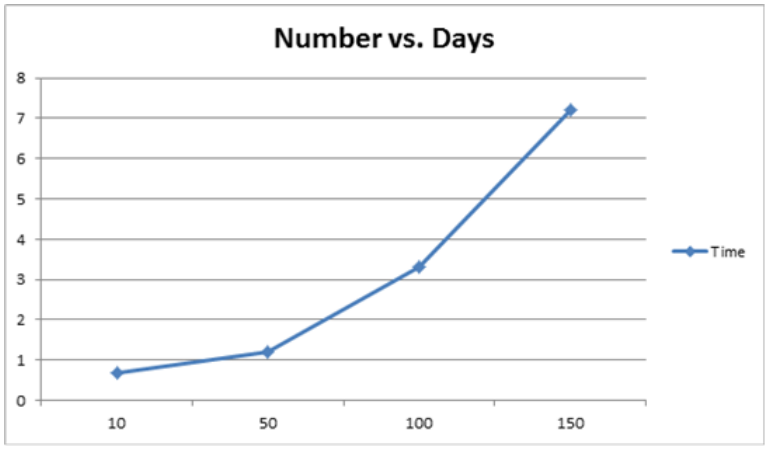


Figure 54: Running times vs. numbers of universities

Chapter 6

Organizing and Storing Extracted University Information

6.1 Information Networking Model

The Information Networking Model is proposed in [53] to model complex relationships. Real world objects have various natural and complex relationships with each other and via these relationships, objects play various roles that form their context and then have the corresponding context-dependent properties. In an Information Networking Model, we can benefit from the following two aspects:

1. Every object is uniquely identified with its object identifier and is associated with exactly one instance that contains complete information about this object via all kinds of relationships.

2. Context-dependent access to object properties is straightforward and the evolutionary, dynamic and many-faceted nature of objects can be naturally reflected.

6.1.1 Schema

```
create class University {Campus} [
        @founded:Int,
        @webSite:url,
        $@type:{Medical, Comprehensive, Teaching},
        @postcode:string,
        normal address: Region(inverse universities),
        contain academics (inverse at) *: AcademicsDepartment,
        role President[
                @office:string,
                @"office hours" *: [@start:string,@end:string],
                role_based (@startDate:date) (inverse position): Person(inverse worksIn)
        ]
]
```

Figure 55: This is a sample schema file using the INM definition language.

Figure 55 illustrates segment of a sample INM schema file for the university class. Major features of Information Networking Model are captured in this minimum example.

Class University is a class defined using the create statement.

Subclass Campus is a subclass of class University.

Simple Attribute Founded, webSite, postcode are all simple attributes distinguished by the symbol @.

Data Type Many predefined data types are present including Int, string, url.

Complex Attribute The office hours attribute falls into this category.

Enum Type Some attributes preceded by the symbol $ can take enum values.

Normal Relationship Address is treated as a normal relationship, whose target is a Region object. Inversely, a Region object has a relationship Universities whose target is a set of University objects.

Contain Relationship It is used to capture the hierarchical relationship. For example, we can use it to express that a university contains many academic departments.

Role Relationship There is a role relationship called President in the schema. Such relationships can have role-based attributes like office and office hours.

Context Information The target of a role relationship is always a Person object. Context information such as where the person works and what the person's position is maintained in the model.

6.1.2 Instance

```
insert University "Carleton University" [
        @founded: 1942,
        @webSite: "WWW.carleton.ca",
        @type: Comprehensive,
        @postcode: "K1S 5B6",
        address: Ottawa,
        academics: [
                "Arts and Social Sciences",
                "Engineering and Design",
                "Graduate & Postdoctoral Affairs",
                "Public Affairs",
                "Science",
                "The Sprott School of Business"
        ],
        President[
                office: "503 Tory Building",
                "office hours": [@start:"9am", @end:"5pm"],
                @startDate: "July 1, 2008"
        ]: "Roseann O'Reilly Runte"
]
```

Figure 56: This is an instance file in INM which captures the basic information about Carleton University.

Figure 56 shows a minimum instance file which corresponds to the schema above. In our system these insert statements will be automatically generated based on the predefined university schema from the extracted university information.

6.1.3 Query

We demonstrate the powerful features of the INM query language by walking through some interesting examples.

1. We want the list of all universities in the greater Chicago area. A sample query statement can be constructed as follows:

query University $X/address: Chicago construct $X[];

The keyword query and construct are reserved in the INM query language to construct query statements. The symbol $ is used to declare a logical variable. In this example, the variable $X is modified by University class, which means $X is mapped to some University instance. The slash is used to select relationships under a University object. In this example, it is a normal relationship called address. The construct part of the statement is intended to describe the format of the result to be returned. In the example, it will return all University instances that satisfy the condition that they are located in the Chicago area.

2. We want to know which university a professor named "Jiawei Han" works for. A sample query statement can be constructed as follows:

query University $X/Professor:* "Jiawei Han" construct $X[];

The variable $X is declared as a University object. All University instances that have "Jiawei Han" as a professor will be returned.

3. We want to know the total number of professors whose research is related to "Data Mining".

query Professor $X/research:"Data Mining" construct count (<$X>);

This query will request the total number of professors who do research in data mining. The count function is similarly defined as in other query languages such SQL.

6.2 System Demonstration

We organize the extracted information of each university based on a schema generalized from our investigation and insert the organized information into an INM database. Through the INM search engine website, we demonstrate our work.

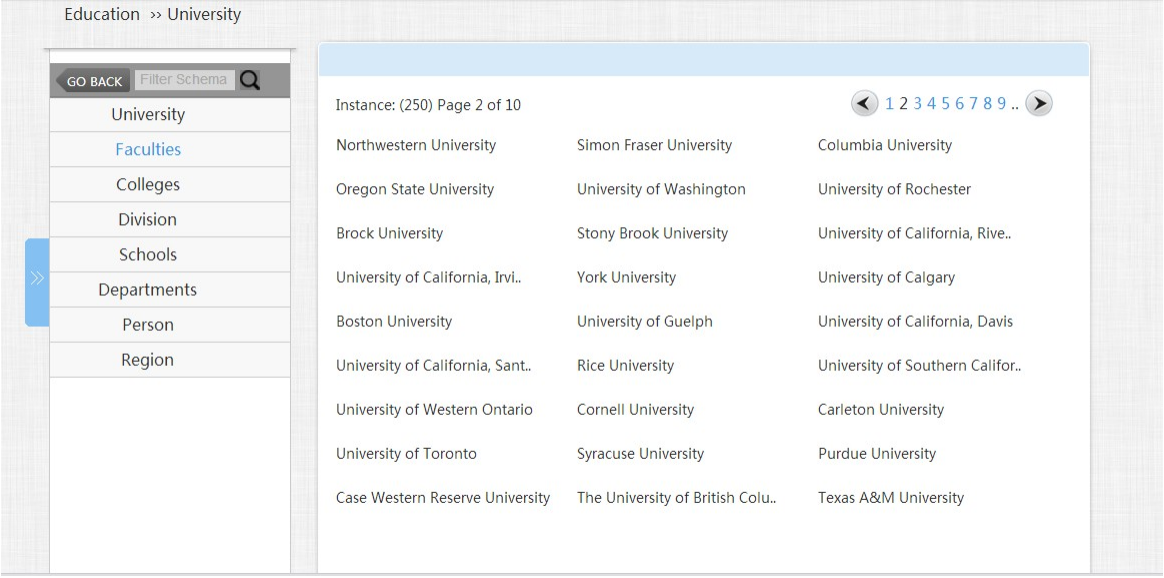


Figure 57: The directory page of all universities.

Right now, we are able to extract 250 universities out of 310 university websites. From Figure 57, we can see the directory of all universities stored in the database. We click on a university named "Carleton University" and end up in a page shown in Figure 58. From that page, we can see general information and faculty information for Carleton University.

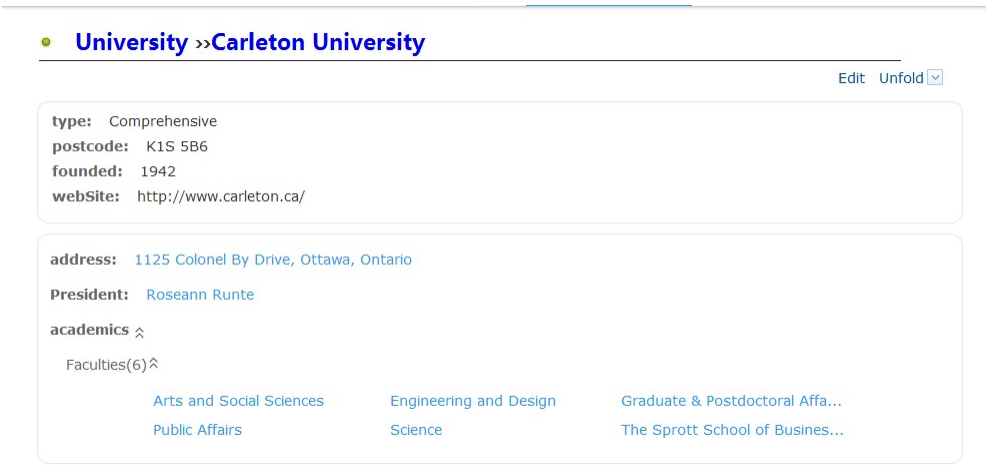


Figure 58: General information and faculty information for Carleton University.

Now we click on one of the faculties named Science and end up with what we can see in Figure 59. From Figure 59, we can see department information for Faculty of Science.

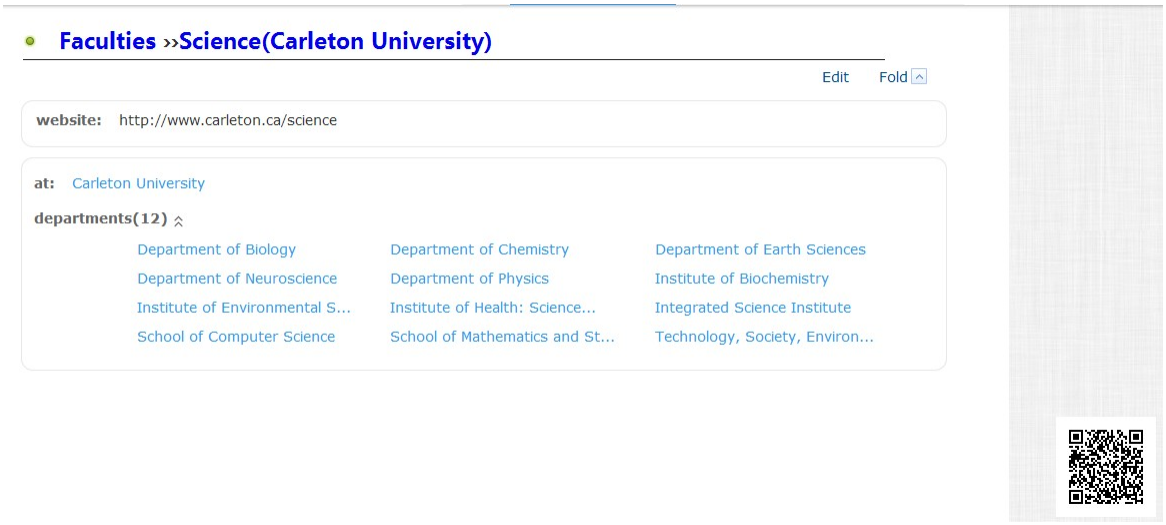


Figure 59: Department information under Faculty of Science at Carleton University.

Clicking on a department named School of Computer Science, we enter the detailed information page for School of Computer Science as shown in Figure 60.

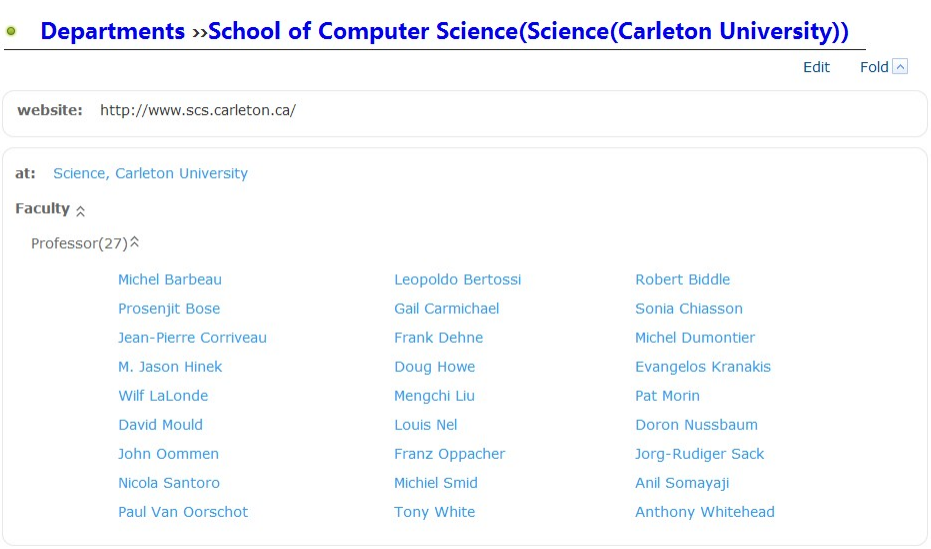


Figure 60: Information for School of Computer Science at Carleton University.

In Figure 60, we can see the list of all faculty members in School of Computer Science.

Figure 61: Information for faculty member Mengchi Liu in School of Computer Science.

As we click on one faculty member named Mengchi Liu, we enter the detailed information page for faculty member Mengchi Liu as shown in Figure 61. Besides navigating through the website like this, we can directly type the faculty member's name in the search box and end up in the same page.

Finally, we show some interesting queries. The first query is that we want to retrieve all universities located in Ottawa. Figure 62 illustrates the search result. The second query is that we want to know which university a person named "Mengchi Liu" works for. We just type in the search box as shown in Figure 63 and the page in Figure 58 will show up as the result. The third query is that we want to find a person named "Bing Liu" under computer science department. This query is useful because we have found more than one "Bing Liu" in our database. Now we construct a query to retrieve the specific "Bing Liu" we want. Figure 64 and 65 show the query statement and the corresponding result respectively.

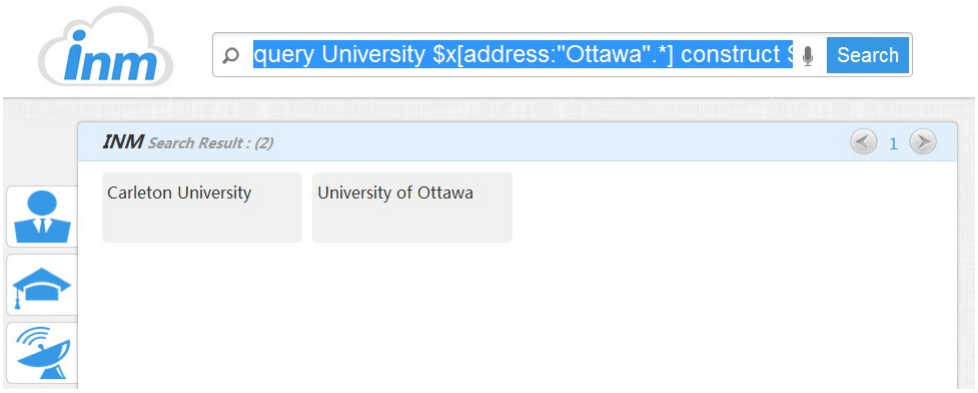


Figure 62: Query for universities in Ottawa.

Figure 63: Query for the university Mengchi Liu works for.

Figure 64: Query for a specific person under a department.

Figure 65: Query result for a person under a department.

Chapter 7

Future Work

The working system is able to extract information from both Canadian and US university websites in a fully automatic way. On one hand, the experimental results demonstrate promising performance for the system. On the other hand, analysis of the results exposes some limitations of the current implementation. In this chapter, we first discuss immediate directions for improving the system. Then we outline our agenda for the future.

7.1 Immediate Directions

For the division list and unit list extraction framework, there are three things we can do to improve the outcome. First, we can further improve the way we generate candidate lists. The current algorithm generates candidate lists at the entire page level. We can divide the page into several information blocks based on containing parent tags and separator tags such as the h tags and hr tag. Then we modify our algorithm to generate candidate lists only within each information block. This modification potentially enables us to remove some restrictions on candidate list generation. For example, there are tiled lists where the first row is not horizontally aligned and nested lists where each item is not vertically aligned. The alignment requirement in

the current algorithm makes it impossible for such lists to be extracted. By confining candidate list generation within an information block, we can consider removing these requirements to accommodate more cases. Second, we can further improve the way we build keyword dictionaries. The current approach is mainly example-based. Although we are able to build a fairly comprehensive dictionary through extensive testing, the potential lack of some keywords can result in the missing of some division or unit information. Another defect of our keyword dictionaries is that these keywords do not form a hierarchy as they should. For example, when we extract units from a division of engineering, we should only consider those keywords existing in engineering related units. Building such an enhanced keyword dictionary with hierarchy can further improve the result. Third, we can further improve the way we process web pages. We do not need to change the algorithm but just switch to more powerful libraries. The support for web page rendering and especially Javascript interpretation is relatively limited in pure Java. We plan to re-implement the page rendering part using one of the major layout engines (e.g., that of Internet Explorer), which is much more reliable in terms of retrieving the right visual information and is capable of handling Javascript-generated content.

For faculty list extraction, there are two scenarios we need to address. The first scenario is where one faculty member in the middle of the list has much less information than others in the list. In that scenario, the tag tree of that faculty member is no longer similar to those of others. As a result, we get two separated faculty lists with that faculty member missing. We need to identify the missing faculty member and combine it with the two separated lists to make the complete faculty list. The second scenario is where each repeated unit contains multiple faculty members and the very last repeated unit contains fewer faculty members than previous units. In that scenario, the tag tree of the last repeated unit is no longer similar to those of

previous units. As a result, the faculty members in the last repeated unit cannot be retrieved as part of the faculty list. We need to check whether the block following the extracted faculty list contains more faculty members and add the additional faculty members to the faculty list if there are any. To resolve these two problems, we will stick to the DOM-based approach for now since the tree alignment algorithm is good for individual faculty attribute extraction. In case it does not work, we will try to incorporate visual features to overcome the limitations. In particular, we can use the faculty list identification algorithm to extract faculty name and faculty homepage and combine visual-based deep web data extraction techniques [17] to identify other attributes in the neighborhood of the faculty name. In the worst case, we can even give up the remaining information in the faculty list and directly extract faculty attributes from their homepages.

7.2 Future Agenda

After building a framework for extracting divisions, units, faculty lists, we first plan to extract more information including research interest, education background, courses, publications, students from faculty member's homepage. This step forward will make our system become very useful. Then, we will try to enhance our system to accommodate more American universities. Finally, we want to extract publication information of each faculty member from external digital libraries and integrate the information into existing databases.

Chapter 8

Conclusion

We manage to build a fully automatic information extraction system in university domain. It is able to extract faculties, colleges, divisions, schools, institutes, departments and faculty members from university websites. We organize and store the extracted information in a database to provide search functions. To search for information about these academic units and faculty members, our search engine has several advantages over traditional search engines: 1) we can obtain more precise information using expressive queries 2) we can specify the order in which we want the information 3) programs or machines are able to process the information directly 4) little irrelevant information will be returned.

Main contributions of this thesis are as follows:

1. We build an ontology which works for top American universities. The ontology can potentially be extended to accommodate hundreds of American universities.

2. We propose a visual-based list extraction framework which overcomes the limitations of existing methods.

3. We demonstrate that it is promising to extract the university domain using a top-down approach.

Appendices

Appendix A

Table 22: Division List Construction Universities

McGill University	http://www.mcgill.ca/
The University of British Columbia	http://www.ubc.ca/
University of Toronto	http://www.utoronto.ca/
Queen's University	http://www.queensu.ca/
University of Alberta	http://www.ualberta.ca/
McMaster University	http://www.mcmaster.ca/
Dalhousie University	http://www.dal.ca/
University of Calgary	http://www.ucalgary.ca/
University of Saskatchewan	http://www.usask.ca/
University of Ottawa	http://www.uottawa.ca/welcome.html
University of Western Ontario	http://www.uwo.ca/
University of Manitoba	http://umanitoba.ca/
Simon Fraser University	http://www.sfu.ca/
University of Victoria	http://www.uvic.ca/
University of Waterloo	http://uwaterloo.ca/
University of New Brunswick	http://www.unb.ca/
University of Guelph	http://www.uoguelph.ca/
Carleton University	http://www.carleton.ca/
Memorial University of Newfoundland	http://www.mun.ca/
York University	http://www.yorku.ca/web/index.htm
University of Regina	http://www.uregina.ca/
University of Windsor	http://www.uwindsor.ca/
Wilfrid Laurier University	http://www.wlu.ca/

Ryerson University	http://www.ryerson.ca/index.html
Concordia University	http://www.concordia.ca/
Brock University	http://www.brocku.ca/
Carnegie Mellon University	http://www.cmu.edu/index.shtml
Massachusetts Institute of Technology	http://web.mit.edu/
University of California, Berkeley	http://berkeley.edu/
Stanford University	http://www.stanford.edu/
Cornell University	http://www.cornell.edu/
University of Illinois at Urbana-Champaign	http://illinois.edu/
University of Washington	http://www.washington.edu/
Princeton University	http://www.princeton.edu/main/
University of Texas at Austin	http://www.utexas.edu/
Georgia Institute of Technology	http://www.gatech.edu/
California Institute of Technology	http://www.caltech.edu/
University of Wisconsin-Madison	http://www.wisc.edu/
University of Michigan	http://www.umich.edu/
University of California, San Diego	http://ucsd.edu/
University of Maryland, College Park	http://www.umd.edu/
University of California, Los Angeles	http://www.ucla.edu/
Columbia University	http://www.columbia.edu/
University of Pennsylvania	http://www.upenn.edu/
Harvard University	http://www.harvard.edu/
University of Massachusetts Amherst	http://www.umass.edu/
Brown University	http://www.brown.edu/
Yale University	http://www.yale.edu/
Purdue University	http://www.purdue.edu/
University of Southern California	http://www.usc.edu/
Rice University	http://www.rice.edu/

University of North Carolina at Chapel Hill	http://www.unc.edu/index.htm
Duke University	http://www.duke.edu/
University of Virginia	http://www.virginia.edu/
New York University	http://www.nyu.edu/
Rutgers University	http://www.rutgers.edu/
Pennsylvania State University	http://www.psu.edu/
Johns Hopkins University	http://www.jhu.edu/
University of California, Irvine	http://www.uci.edu/
Ohio State University	http://www.osu.edu/
Northwestern University	http://www.northwestern.edu/
University of Minnesota, Twin Cities	http://www1.umn.edu/twincities/index.html
University of Chicago	http://www.uchicago.edu/index.shtml
University of California, Santa Barbara	http://www.ucsb.edu/
University of California, Davis	http://www.ucdavis.edu/
University of Florida	http://www.ufl.edu/
University of Utah	http://www.utah.edu/
Washington University in St. Louis	http://wustl.edu/
University of Colorado Boulder	http://www.colorado.edu/
Dartmouth College	http://www.dartmouth.edu/
Virginia Tech	http://www.vt.edu/
Stony Brook University	http://www.stonybrook.edu/sb/
North Carolina State University	http://www.ncsu.edu/
Rensselaer Polytechnic Institute	http://rpi.edu/
Texas A&M University	http://www.tamu.edu/
University of Arizona	http://www.arizona.edu/
University of Rochester	http://www.rochester.edu/
Boston University	http://www.bu.edu/
University of California, Riverside	http://www.ucr.edu/
Arizona State University	http://www.asu.edu/

Indiana University Bloomington	http://www.iub.edu/
University of Pittsburgh	http://www.pitt.edu/
University of California, Santa Cruz	http://www.ucsc.edu/
University of Illinois at Chicago	http://www.uic.edu/uic/
Michigan State University	http://www.msu.edu/
Vanderbilt University	http://www.vanderbilt.edu/
Northeastern University	http://www.northeastern.edu/
University at Buffalo	http://www.buffalo.edu/
Syracuse University	http://www.syr.edu/
University of Notre Dame	http://nd.edu/
University of Tennessee	http://www.utk.edu/
George Mason University	http://www.gmu.edu/
University of Oregon	http://www.uoregon.edu/
Oregon State University	http://oregonstate.edu/
Case Western Reserve University	http://www.case.edu/
University of Iowa	http://www.uiowa.edu/
Tufts University	http://www.tufts.edu/
Wake Forest University	http://www.wfu.edu/
Emory University	http://www.emory.edu/home/index.html
Brandeis University	http://www.brandeis.edu/

Table 24: Division List Testing Universities

Abilene Christian University	http://www.acu.edu/
Academy of Art University	http://www.academyart.edu/
Alabama State University	http://www.alasu.edu/
Alaska Pacific University	http://www.alaskapacific.edu/
Alcorn State University	http://www.alcorn.edu
Alfred University	http://www.alfred.edu/
Allen University	http://www.allenuniversity.edu
American Military University	http://www.apus.edu/amu
American Public University	http://www.apus.edu/apu

American University	http://www.american.edu/
Anderson University	http://www.anderson.edu/
Andrews University	http://www.andrews.edu/
Angelo State University	http://www.angelo.edu/
Appalachian State University	http://www.appstate.edu/
Arizona State University West	http://www.west.asu.edu/
Arkansas Tech University	http://www.atu.edu/
Ashford University	http://www.ashford.edu
Ashland University	http://www.ashland.edu/
Auburn University, Montgomery	http://www.aum.edu/
Auburn University	http://www.auburn.edu/
Aurora University	http://www.aurora.edu/
Austin Peay State University	http://www.apsu.edu/
Azusa Pacific University	http://apu.edu/
Bastyr University	http://www.bastyr.edu/
Baylor University	http://www.baylor.edu/
Bellarmine University	http://www.bellarmine.edu/
Bemidji State University	http://www.bemidjistate.edu
Benedictine University	http://www.ben.edu/
Bowling Green State University	http://www.bgsu.edu/
Bradley University	http://www.bradley.edu/
Butler University	http://www.butler.edu/
California Baptist University	http://www.calbaptist.edu
California State University, Chico	http://www.csuchico.edu/
California State University, Dominguez Hills	http://www.csudh.edu/
California State University, Fresno	http://www.csufresno.edu/
California State University, Long Beach	http://www.csulb.edu/
California State University, Los Angeles	http://www.calstatela.edu/
California State University, Sacramento	http://www.csus.edu/

California State University, San Bernardino	http://www.csusb.edu/
California State University, San Marcos	http://www.csusm.edu/
Campbell University	http://www.campbell.edu/
Campbellsville University	http://www.campbellsville.edu/
Capital University	http://www.capital.edu/
Cedarville University	http://www.cedarville.edu/
Chapman University	http://www.chapman.edu
Clarion University	http://www.clarion.edu/
Clarkson University	http://www.clarkson.edu/
Clemson University	http://www.clemson.edu/
Cleveland State University	http://www.csuohio.edu/
Coastal Carolina University	http://www.coastal.edu/
Colorado State University	http://www.colostate.edu/
Cornerstone University	http://www.cornerstone.edu/
Creighton University	http://www.creighton.edu/
Dallas Baptist University	http://www.dbu.edu/
DePaul University	http://www.depaul.edu/
Dickinson State University	http://www.dsu.nodak.edu/
Dominican University of California	http://www.dominican.edu/
Drexel University	http://www.drexel.edu/
Duquesne University	http://www.duq.edu/
East Carolina University	http://www.ecu.edu/
East Central University, Ada Oklahoma	http://www.ecok.edu/
East Tennessee State University	http://www.etsu.edu
Eastern Michigan University	http://www.emich.edu/
Elon University	http://www.elon.edu/
Evangel University	http://www.evangel.edu/
Everglades University	http://www.evergladesuniversity.edu
Faulkner University	http://www.faulkner.edu/
Fayetteville State University	http://www.uncfsu.edu/
Ferris State University	http://www.ferris.edu/
Finlandia University	http://www.finlandia.edu
Florida Gulf Coast University	http://www.fgcu.edu/

Florida International University	http://www.fiu.edu/
Francis Marion University	http://www.fmarion.edu/
Freed-Hardeman University	http://www.fhu.edu/
George Fox University	http://www.georgefox.edu/
George Washington University	http://www.gwu.edu
Georgetown University	http://www.georgetown.edu/
Georgia Southwestern State University	http://www.gsw.edu/
Georgia State University	http://www.gsu.edu/
Globe University	http://www.globeuniversity.edu/
Gonzaga University	http://www.gonzaga.edu/
Grambling State University	http://www.gram.edu
Grand Canyon University	http://www.grand-canyon.edu
Hardin-Simmons University	http://www.hsutx.edu/
Hofstra University	http://www.hofstra.edu/
Huntington University	http://www.huntington.edu
Idaho State University	http://www.isu.edu/
Indiana State University	http://www.indstate.edu/
Indiana University - Purdue University, Indianapolis	http://www.iupui.edu/
Indiana University Northwest	http://www.iun.edu
Indiana University Southeast	http://www.ius.edu
Indiana University at South Bend	http://www.iusb.edu/
Indiana University of Pennsylvania	http://www.iup.edu/
Iowa State University	http://www.iastate.edu/
Jackson State University	http://www.jsums.edu
Johnson C. Smith University	http://www.jcsu.edu/
Kansas State University	http://www.ksu.edu/
Kennesaw State University	http://www.kennesaw.edu/
Kentucky State University	http://www.kysu.edu/
Lake Superior State University	http://www.lssu.edu/
Lamar University	http://www.lamar.edu/
Lander University	http://www.lander.edu/

Lawrence Technological University	http://www.ltu.edu/
Liberty University	http://www.liberty.edu/
Lincoln University, Jefferson City Missouri	http://www.lincolnu.edu/
Lincoln University, San Francisco California	http://www.lincolnuca.edu/
Lipscomb University	http://www.lipscomb.edu/
Louisiana State University at Baton Rouge	http://www.lsu.edu/
Loyola Marymount University	http://www.lmu.edu/
Loyola University, Chicago	http://www.luc.edu/
Loyola University, New Orleans	http://www.loyno.edu/
Lynn University	http://www.lynn.edu/
Marquette University	http://www.mu.edu/
Marshall University	http://www.marshall.edu/
Marymount University	http://www.marymount.edu/
Mayville State University	http://www.masu.nodak.edu/
McMurry University	http://www.mcm.edu/
Mercer University	http://www.mercer.edu/
Miami University of Ohio	http://www.muohio.edu/
Michigan Technological University	http://www.mtu.edu/
Middle Tennessee State University	http://www.mtsu.edu/
Minnesota State University Mankato	http://www.mnsu.edu/
Minnesota State University Moorhead	http://www.mnstate.edu/
Mississippi State University	http://www.msstate.edu/
Mississippi University for Women	http://www.muw.edu/
Missouri State University	http://www.missouristate.edu
Montana State University-Billings	http://www.msubillings.edu/
Montana State University-Bozeman	http://www.montana.edu/
Montclair State University	http://www.montclair.edu/
Naropa University	http://www.naropa.edu/

National American University	http://www.national.edu/
Nebraska Wesleyan University	http://www.nebrwesleyan.edu/
New Mexico State University	http://www.nmsu.edu/
Nicholls State University	http://www.nicholls.edu/
Norfolk State University	http://www.nsu.edu/
North Carolina Central University	http://www.nccu.edu/
North Greenville University	http://www.ngu.edu
Northern Arizona University	http://www.nau.edu
Northern Kentucky University	http://www.nku.edu
Northwest Missouri State University	http://www.nwmissouri.edu/
Ohio Northern University	http://www.onu.edu/
Ohio University	http://www.ohiou.edu/
Oklahoma Christian University	http://www.oc.edu/
Oral Roberts University	http://www.oru.edu/
Our Lady of the Lake University	http://www.ollusa.edu/
Pacific University	http://www.pacificu.edu/
Pennsylvania State University at Harrisburg	http://www.hbg.psu.edu/
Pepperdine University	http://www.pepperdine.edu/
Philadelphia Biblical University	http://cairn.edu/
Pittsburg State University	http://www.pittstate.edu/
Portland State University	http://www.pdx.edu/
Prairie View A & M University	http://www.pvamu.edu/
Rockhurst University	http://www.rockhurst.edu/
Sacred Heart University	http://www.sacredheart.edu/
Saint Cloud State University	http://www.stcloudstate.edu/
Saint John's University, Jamaica New York	http://www.stjohns.edu/
Saint Martin's University	http://www.stmartin.edu/

Sam Houston State University	http://www.shsu.edu/
San Diego State University	http://www.sdsu.edu/
San Jose State University	http://www.sjsu.edu/
Santa Clara University	http://www.scu.edu/
Seattle University	http://www.seattleu.edu/
Seton Hall University	http://www.shu.edu/
Shawnee State University	http://www.shawnee.edu/
Shenandoah University	http://www.su.edu/
Shippensburg University of Pennsylvania	http://www.ship.edu/
Sonoma State University	http://www.sonoma.edu/
South Dakota State University	http://www.sdstate.edu/
Southeast Missouri State University	http://www.semo.edu/
Southern Arkansas University	http://www.saumag.edu/
Southern Connecticut State University	http://www.southernct.edu
Southern Illinois University at Carbondale	http://www.siu.edu/
Southern Methodist University	http://www.smu.edu/
Southern Oregon University	http://www.sou.edu
Southern University, Baton Rouge	http://www.subr.edu/
Southern University, New Orleans	http://www.suno.edu/
Southwestern Assemblies of God University	http://www.sagu.edu/
Southwestern Oklahoma State University	http://www.swosu.edu/
Spalding University	http://www.spalding.edu/
State University of New York College Maritime College at Fort Schuyler	http://www.sunymaritime.edu/
State University of New York College at Buffalo (Buffalo State College)	http://www.buffalostate.edu

State University of New York College at Fredonia	http://www.fredonia.edu/
State University of New York College at Geneseo	http://mosaic.cc.geneseo.edu/
State University of New York College at New Paltz	http://www.newpaltz.edu/
State University of New York College of Agriculture and Technology, Morrisville	http://www.morrisville.edu/
State University of New York at Albany	http://www.albany.edu/
State University of New York at Binghamton	http://www.binghamton.edu/
State University of New York at Buffalo	http://www.buffalo.edu/
State University of New York at Oswego	http://www.oswego.edu/
Stetson University	http://www.stetson.edu/
Stratford University	http://www.stratford.edu
Tarleton State University	http://www.tarleton.edu/
Temple University	http://www.temple.edu/
Tennessee State University	http://www.tnstate.edu/
Texas A&M International University	http://www.tamiu.edu/
Texas A&M University, Corpus Christi	http://www.tamucc.edu/
Texas A&M University, Kingsville	http://www.tamuk.edu/
Texas Christian University	http://www.tcu.edu/
Texas Woman's University	http://www.twu.edu/
The Catholic University of America	http://www.cua.edu/

Table 26: Unit List Construction Divisions

Faculty of Agricultural and Environmental Sciences	http://www.mcgill.ca/macdonald/
Faculty of Dentistry	http://www.mcgill.ca/dentistry/

Faculty of Education	http://www.mcgill.ca/education/
Faculty of Engineering	http://www.mcgill.ca/engineering/
Faculty of Law	http://www.mcgill.ca/law/
Desautels Faculty of Management	http://www.mcgill.ca/desautels/
Faculty of Medicine	http://www.mcgill.ca/medicine/
Schulich School of Music	http://www.mcgill.ca/music/
Faculty of Religious Studies	http://www.mcgill.ca/religiousstudies/
Faculty of Science	http://www.mcgill.ca/science/
Applied Science, Faculty of	http://www.apsc.ubc.ca/
Architecture and Landscape Architecture, School of	http://www.sala.ubc.ca/
Arts, Faculty of	http://www.arts.ubc.ca/
Audiology and Speech Sciences, School of	http://www.audiospeech.ubc.ca/
Business, Sauder School of	http://www.sauder.ubc.ca/
Community and Regional Planning, School of	http://www.scarp.ubc.ca/
Dentistry, Faculty of	http://www.dentistry.ubc.ca/
Education, Faculty of	http://www.educ.ubc.ca/
Environmental Health, School of	http://www.soeh.ubc.ca/
Forestry, Faculty of	http://www.forestry.ubc.ca/
Health Disciplines, College of	http://www.chd.ubc.ca/
Journalism, School of	http://www.journalism.ubc.ca/
Kinesiology, School of	http://www.kin.ubc.ca/
Land and Food Systems, Faculty of	http://www.landfood.ubc.ca/
Law, Faculty of	http://www.law.ubc.ca/
Library, Archival and Information Studies, School of	http://www.slais.ubc.ca/
Medicine, Faculty of	http://www.med.ubc.ca/
Music, School of	http://www.music.ubc.ca/
Nursing, School of	http://www.nursing.ubc.ca/
Population and Public Health, School of	http://www.spph.ubc.ca/
Pharmaceutical Sciences, Faculty of	http://www.pharmacy.ubc.ca/
Science, Faculty of	http://www.science.ubc.ca/

Social Work, School of	http://www.socialwork.ubc.ca/
Applied Science and Engineering, Faculty of	http://www.engineering.utoronto.ca/
Architecture, Landscape, and Design, John H. Daniels Faculty of	http://www.daniels.utoronto.ca/
Arts and Science, Faculty of	http://www.artsci.utoronto.ca/
Dentistry, Faculty of	http://www.dentistry.utoronto.ca/
Education, Ontario Institute for Studies in	http://www.oise.utoronto.ca/
Forestry, Faculty of	http://www.forestry.utoronto.ca/
Information, Faculty of	http://www.ischool.utoronto.ca/
Kinesiology and Physical Education, Faculty of	http://www.utoronto.ca/physical
Law, Faculty of	http://www.law.utoronto.ca/
Management, Joseph L. Rotman School of	http://www.rotman.utoronto.ca/index.html
Medicine, Faculty of	http://medicine.utoronto.ca/
Music, Faculty of	http://www.music.utoronto.ca/
Nursing, Lawrence S. Bloomberg Faculty of	http://bloomberg.nursing.utoronto.ca/
Pharmacy, Leslie L. Dan Faculty of	http://www.utoronto.ca/pharmacy/
Public Health, Dalla Lana School of	http://www.dlsph.utoronto.ca/
Social Work, Factor-Inwentash Faculty of	http://www.socialwork.utoronto.ca/
Faculty of Arts and Science	http://www.queensu.ca/artsci/
Faculty of Education	http://educ.queensu.ca/
Faculty of Engineering and Applied Science	http://engineering.queensu.ca/
Faculty of Health Sciences	http://healthsci.queensu.ca/
Faculty of Law	http://law.queensu.ca/
School of Business	http://business.queensu.ca/index.php
College of Arts	http://www.uoguelph.ca/arts
College of Biological Science	http://www.uoguelph.ca/cbs/
College of Management & Economics	http://www.uoguelph.ca/cme/
College of Physical & Engineering Science	http://www.uoguelph.ca/cpes/

College of Social & Applied Human Sciences	http://www.csahs.uoguelph.ca/
Ontario Agricultural College	http://www.oac.uoguelph.ca/
Ontario Veterinary College	http://www.ovc.uoguelph.ca/
School of Architecture and Planning	http://sap.mit.edu
School of Engineering	http://engineering.mit.edu/
School of Humanities, Arts, and Social Sciences	http://shass.mit.edu/
Sloan School of Management	http://mitsloan.mit.edu/
School of Science	http://web.mit.edu/science/
Whitaker College of Health Sciences and Technology	http://hst.mit.edu/
Letters & Science, College of	http://ls.berkeley.edu/
Business, Haas School of	http://haas.berkeley.edu/
Chemistry, College of	http://chemistry.berkeley.edu/
Engineering, College of	http://coe.berkeley.edu/
Environmental Design, College of	http://ced.berkeley.edu/
Information, School of	http://www.ischool.berkeley.edu/
Journalism, Graduate School of	http://journalism.berkeley.edu/
Law, School of	http://www.law.berkeley.edu/
Natural Resources, College of	http://cnr.berkeley.edu/
Optometry, School of	http://optometry.berkeley.edu/
Public Health, School of	http://sph.berkeley.edu/
Public Policy, Richard & Rhoda Goldman School of	http://gspp.berkeley.edu/
Social Welfare, School of	http://socialwelfare.berkeley.edu/
Business, Graduate School of	http://www.gsb.stanford.edu/
Earth Sciences, School of	https://pangea.stanford.edu/
Education, School of	https://ed.stanford.edu/
Engineering, School of	http://soe.stanford.edu/
Humanities & Sciences, School of	http://www.stanford.edu/dept/humsci/
Law School	http://www.law.stanford.edu/

Medicine, School of	http://med.stanford.edu/
College of Agriculture and Life Sciences	http://www.cals.cornell.edu/
College of Architecture, Art, and Planning	http://www.aap.cornell.edu/
College of Arts and Sciences	http://as.cornell.edu/
College of Engineering	http://www.engineering.cornell.edu/
School of Hotel Administration	http://www.hotelschool.cornell.edu/
College of Human Ecology	http://www.human.cornell.edu/
School of Industrial and Labor Relations (ILR)	http://www.ilr.cornell.edu/
The Faculty of Computing and Information Science	http://www.cis.cornell.edu/
Cornell Law School	http://www.lawschool.cornell.edu/
College of Veterinary Medicine	http://www.vet.cornell.edu/
Harvard Divinity School	http://www.hds.harvard.edu/
School of Engineering and Applied Sciences	http://seas.harvard.edu/
Graduate School of Arts & Sciences	http://gsas.harvard.edu/
Harvard Medical School	http://hms.harvard.edu/
Harvard Business School	http://www.hbs.edu/
Faculty of Arts & Sciences	http://fas.harvard.edu/
Graduate School of Design	http://www.gsd.harvard.edu/
Harvard Graduate School of Education	http://www.gse.harvard.edu/
Harvard Kennedy School	http://www.hks.harvard.edu/
Harvard Law School	http://www.law.harvard.edu/index.html
Harvard School of Public Health	http://www.hsph.harvard.edu/
College of Agriculture	http://www.agriculture.purdue.edu
College of Education	http://www.education.purdue.edu/
College of Engineering	https://engineering.purdue.edu/Engr/
College of Health and Human Sciences	http://www.purdue.edu/hhs
College of Liberal Arts	http://www.cla.purdue.edu
Krannert School of Management	http://www.krannert.purdue.edu

College of Pharmacy	http://www.pharmacy.purdue.edu/
College of Science	http://www.science.purdue.edu
College of Technology	http://www.tech.purdue.edu/
College of Veterinary Medicine	http://www.vet.purdue.edu
Trinity College of Arts & Sciences	http://trinity.duke.edu/
Divinity School	http://www.divinity.duke.edu/
Fuqua School of Business	http://www.fuqua.duke.edu/
School of Law	http://www.law.duke.edu/
School of Medicine	http://medschool.duke.edu/
Nicholas School of the Environment	http://www.nicholas.duke.edu/
School of Nursing	http://nursing.duke.edu/
Pratt School of Engineering	http://www.pratt.duke.edu/
Sanford School of Public Policy	http://sanford.duke.edu/
Arts	http://www.arts.uci.edu/
Biological Sciences	http://www.bio.uci.edu/
Business	http://merage.uci.edu/
Education	http://gse.uci.edu/
Engineering	http://www.eng.uci.edu/
Humanities	http://www.humanities.uci.edu/
Information & Computer Sciences	http://ics.uci.edu/
Law	http://law.uci.edu/
Medicine	http://som.uci.edu/
Nursing Science	http://www.nursing.uci.edu/
Pharmaceutical Sciences	http://pharmsci.uci.edu/
Physical Sciences	http://www.physsci.uci.edu/
Public Health	http://publichealth.uci.edu/
Social Ecology	http://socialecology.uci.edu/
Social Sciences	http://www.socsci.uci.edu/
The Betty Irene Moore School of Nursing	http://www.ucdmc.ucdavis.edu/nursing/
Graduate School of Management	http://www.gsm.ucdavis.edu/
School of Education	http://education.ucdavis.edu/
College of Agricultural and Environmental Sciences	http://caes.ucdavis.edu/

College of Biological Sciences	http://biosci.ucdavis.edu/index_js.html
College of Engineering	http://engineering.ucdavis.edu/
College of Letters and Science	http://www.ls.ucdavis.edu/
School of Law	http://www.law.ucdavis.edu/academics-clinicals/index.html
School of Medicine	http://www.ucdmc.ucdavis.edu/medschool/
School of Veterinary Medicine	http://www.vetmed.ucdavis.edu/students/dvm_program/dvm_curriculum/index.cfm
Architecture + Planning	http://www.arch.utah.edu/
Business	http://www.business.utah.edu/
Dentistry	http://dentistry.utah.edu/
Education	http://www.ed.utah.edu/
Engineering	http://www.coe.utah.edu/
Fine Arts	http://www.finearts.utah.edu/
Health	http://www.health.utah.edu/
Humanities	http://humanities.utah.edu/
Law	http://www.law.utah.edu/
Medicine	http://medicine.utah.edu/
Mines & Earth Sciences	http://www.cmes.utah.edu/
Nursing	http://nursing.utah.edu
Pharmacy	http://www.pharmacy.utah.edu/
Science	http://www.science.utah.edu/
Social & Behavioral Science	http://www.csbs.utah.edu/
Social Work	http://www.socwk.utah.edu/
College of Agriculture and Life Sciences	http://harvest.cals.ncsu.edu/indexmain.cfm
College of Design	http://design.ncsu.edu/
College of Education	http://ced.ncsu.edu/
College of Engineering	http://www.engr.ncsu.edu/
College of Humanities and Social Sciences	http://www.chass.ncsu.edu/index.php
College of Natural Resources	http://natural-resources.ncsu.edu/
Poole College of Management	http://www.mgt.ncsu.edu/index.php
College of Sciences	http://sciences.ncsu.edu/
College of Textiles	http://www.tx.ncsu.edu/

College of Veterinary Medicine	http://www.cvm.ncsu.edu/
Agricultural Sciences	http://agsci.oregonstate.edu/
Business	http://business.oregonstate.edu/
Earth, Ocean & Atmospheric Sciences	http://ceoas.oregonstate.edu/
Education	http://education.oregonstate.edu/
Engineering	http://engr.oregonstate.edu/
Forestry	http://www.cof.orst.edu/
Public Health & Human Sciences	http://health.oregonstate.edu/
Liberal Arts	http://oregonstate.edu/cla/
Pharmacy	http://pharmacy.oregonstate.edu/
Science	http://www.science.orst.edu/
Veterinary Medicine	http://vetmed.oregonstate.edu/
College of Liberal Arts and Sciences	http://www.clas.uiowa.edu/
Tippie College of Business	http://tippie.uiowa.edu/
College of Dentistry	http://www.dentistry.uiowa.edu/
College of Education	http://www.education.uiowa.edu/
College of Engineering	http://www.engineering.uiowa.edu/
College of Law	http://www.law.uiowa.edu/
Carver College of Medicine	http://www.medicine.uiowa.edu/
College of Nursing	http://www.nursing.uiowa.edu/
College of Pharmacy	http://pharmacy.uiowa.edu/
College of Public Health	http://www.public-health.uiowa.edu/

Table 28: Unit List Testing Divisions

DeGroote School of Business	http://www.degroote.mcmaster.ca
Faculty of Engineering	http://www.eng.mcmaster.ca
Faculty of Health Sciences	http://fhs.mcmaster.ca
Faculty of Humanities	http://www.humanities.mcmaster.ca
Faculty of Science	http://www.science.mcmaster.ca
Faculty of Social Sciences	http://www.socsci.mcmaster.ca/
Arts & Science	http://mcmaster.ca/artsci/
Arts & Humanities	http://www.uwo.ca/arts/

Don Wright Faculty of Music	http://www.music.uwo.ca/
Education	http://www.edu.uwo.ca/
Engineering	http://www.eng.uwo.ca/
Health Sciences	http://www.uwo.ca/fhs/
Information & Media Studies	http://www.fims.uwo.ca/index.htm
Law	http://www.law.uwo.ca/
Ivey Business School	http://www.ivey.uwo.ca/
Schulich Medicine & Dentistry	http://www.schulich.uwo.ca/
Science	http://www.uwo.ca/sci/
Social Science	http://www.ssc.uwo.ca/
Applied Health Sciences	https://uwaterloo.ca/applied-health-sciences/
Arts	http://arts.uwaterloo.ca/
Engineering	https://uwaterloo.ca/engineering/
Environment	https://uwaterloo.ca/environment/
Mathematics	http://math.uwaterloo.ca/math/
Science	https://uwaterloo.ca/science/
Arts and Social Sciences	http://www.carleton.ca/fass/
Engineering and Design	http://www.carleton.ca/engineering
Public Affairs	http://www.carleton.ca/fpa
Science	http://www.carleton.ca/science
The Sprott School of Business	http://www.carleton.ca/sprott
Arts & Humanities	http://dah.ucsd.edu/
Biological Sciences	http://biology.ucsd.edu/
Jacobs School of Engineering	http://www.jacobsschool.ucsd.edu/
Physical Sciences	http://physicalsciences.ucsd.edu/
Rady School of Management	http://rady.ucsd.edu/
School of International Relations and Pacific Studies	http://irps.ucsd.edu/
School of Medicine	http://som.ucsd.edu/
Scripps Institution of Oceanography	http://www.sio.ucsd.edu/
Skaggs School of Pharmacy & Pharmaceutical Sciences	http://pharmacy.ucsd.edu/index.shtml
Social Sciences	http://socialsciences.ucsd.edu/

Annenberg School for Communication	http://www.asc.upenn.edu/
Arts & Sciences	http://www.sas.upenn.edu/
Dental Medicine	http://www.dental.upenn.edu/
Design	http://www.design.upenn.edu
Engineering	http://www.seas.upenn.edu/
Graduate School of Education	http://www.gse.upenn.edu/
Law School	http://www.law.upenn.edu/
Nursing	http://www.nursing.upenn.edu/
Perelman School of Medicine	http://www.med.upenn.edu/
Social Policy & Practice	http://www.sp2.upenn.edu/
Veterinary Medicine	http://www.vet.upenn.edu/
The Wharton School	http://www.wharton.upenn.edu/
School of Architecture	http://www.arch.virginia.edu/
College and Graduate School of Arts & Sciences	http://artsandsciences.virginia.edu/
McIntire School of Commerce	http://www.commerce.virginia.edu/
Darden School of Business	http://www.darden.virginia.edu/
Curry School of Education	http://curry.virginia.edu/
School of Engineering and Applied Science	http://www.seas.virginia.edu/
Frank Batten School of Leadership and Public Policy	http://batten.virginia.edu/
Arts	http://www.arts.uci.edu/
Biological Sciences	http://www.bio.uci.edu/
Business	http://merage.uci.edu/
Education	http://gse.uci.edu/
Engineering	http://www.eng.uci.edu/
Humanities	http://www.humanities.uci.edu/
Information & Computer Sciences	http://ics.uci.edu/
Law	http://law.uci.edu/
Medicine	http://som.uci.edu/
Nursing Science	http://www.nursing.uci.edu/
Pharmaceutical Sciences	http://pharmsci.uci.edu/
Physical Sciences	http://www.physsci.uci.edu/
Public Health	http://publichealth.uci.edu/

Social Ecology	http://socialecology.uci.edu/
Social Sciences	http://www.socsci.uci.edu/
Agricultural and Life Sciences	http://cals.ufl.edu/
Business Administration	http://warrington.ufl.edu/
Dentistry	http://www.dental.ufl.edu/
Design, Construction and Planning	http://www.dcp.ufl.edu/
Education	http://www.coe.ufl.edu/
Engineering	http://www.eng.ufl.edu/
Fine Arts	http://www.arts.ufl.edu/
Health and Human Performance	http://www.hhp.ufl.edu/
Journalism and Communications	http://www.jou.ufl.edu/
Law	http://www.law.ufl.edu/
Liberal Arts and Sciences	http://www.clas.ufl.edu/
Medicine	http://www.med.ufl.edu/
Nursing	http://con.ufl.edu/
Pharmacy	http://www.cop.ufl.edu/
Public Health and Health Professions	http://www.hp.ufl.edu/
Veterinary Medicine	http://www.vetmed.ufl.edu/
Bourns College of Engineering (BCOE)	http://www.engr.ucr.edu/
College of Humanities, Arts, & Social Sciences (CHASS)	http://www.chass.ucr.edu/
College of Natural & Agricultural Sciences (CNAS)	http://www.cnas.ucr.edu/
School of Business Administration (SoBA)	http://soba.ucr.edu/
Graduate School of Education (GSOE)	http://www.education.ucr.edu/
School of Medicine	http://medschool.ucr.edu/
Business	http://wpcarey.asu.edu/
Design and the Arts	http://herbergerinstitute.asu.edu
Education	http://education.asu.edu/
Engineering	http://engineer.asu.edu/
Health Solutions	https://chs.asu.edu/
Journalism	http://cronkite.asu.edu

Law	http://www.law.asu.edu/
Nursing and Health Innovation	http://nursingandhealth.asu.edu/
Public Programs	http://copp.asu.edu
Sustainability	http://schoolofsustainability.asu.edu
Technology and Innovation	http://technology.poly.asu.edu/
Division of the Arts	http://arts.ucsc.edu
Division of Humanities	http://humanities.ucsc.edu
Division of Physical & Biological Sciences	http://pbsci.ucsc.edu
Division of Social Sciences	http://socialsciences.ucsc.edu
Jack Baskin School of Engineering	http://soe.ucsc.edu/
College of Arts & Sciences	http://cas.appstate.edu/
College of Fine and Applied Arts	http://www.faa.appstate.edu/
College of Health Sciences	http://www.healthcollege.appstate.edu
Hayes School of Music	http://www.music.appstate.edu/
Reich College of Education	http://www.ced.appstate.edu/
College of Arts and Humanities (CAH)	http://cah.csudh.edu/
College of Business Administration and Public Policy (CBAPP)	http://cbapp.csudh.edu/
College of Education (COE)	http://www.csudh.edu/cps/soe/
College of Natural and Behavioral Sciences (CNBS)	http://www.nbs.csudh.edu/
College of Health, Human Services and Nursing	http://www.csudh.edu/cps/
College of the Arts	http://www.csulb.edu/cota
College of Business Administration	http://www.csulb.edu/cba
College of Education	http://www.ced.csulb.edu/
College of Engineering	http://www.csulb.edu/coe
College of Health & Human Services	http://www.csulb.edu/chhs
College of Liberal Arts	http://www.csulb.edu/cla
College of Natural Sciences & Mathematics	http://www.cnsm.csulb.edu/

Arts & Letters	http://www.csus.edu/al/
Education	http://www.csus.edu/coe/
Natural Sciences & Mathematics	http://www.csus.edu/nsm/
Business Administration	http://www.cba.csus.edu/
Engineering & Computer Science	http://www.ecs.csus.edu/
Social Sciences & Interdisciplinary Studies	http://www.csus.edu/ssis/
College of Business	http://www3.dbu.edu/business/
College of Christian Faith	http://www3.dbu.edu/christian_faith/
College of Education	http://www3.dbu.edu/education/
College of Fine Arts	http://www3.dbu.edu/fine_arts
College of Humanities & Social Sciences	http://www3.dbu.edu/humanities
College of Natural Sciences & Mathematics	http://www3.dbu.edu/math_science
Arts and Sciences	http://www.emich.edu/cas/
Business	http://www.emich.edu/cob/
Education	http://www.emich.edu/coe/
Health and Human Services	http://www.emich.edu/chhs/
Technology	http://www.emich.edu/cot/
Arts and Sciences	http://www.ferris.edu/HTMLS/colleges/artsands/homepage.htm
Business	http://www.ferris.edu/business
Education and Human Services	http://www.ferris.edu/HTMLS/colleges/educatio/homepage.htm
Engineering Technology	http://www.ferris.edu/HTMLS/colleges/technolo/homepage.htm
Health Professions	http://www.ferris.edu/HTMLS/colleges/alliedhe/
Kendall College of Art and Design	http://www.kcad.edu/
Michigan College of Optometry	http://www.ferris.edu/HTMLS/colleges/michopt/homepage.htm
Pharmacy	http://www.ferris.edu/HTMLS/colleges/pharmacy/homepage.htm
Columbian College of Arts and Sciences	http://columbian.gwu.edu/
School of Medicine and Health Sciences	http://smhs.gwu.edu/

GW Law	http://www.law.gwu.edu/
School of Engineering and Applied Science	http://www.seas.gwu.edu/
Graduate School of Education and Human Development	http://gsehd.gwu.edu/
School of Business	http://business.gwu.edu/
Elliott School of International Affairs	http://elliott.gwu.edu/
School of Public Health and Health Services	http://sphhs.gwu.edu/
School of Nursing	http://nursing.gwu.edu/
School of Arts and Letters	http://www.ius.edu/artsandletters/
School of Business	http://www.ius.edu/business/
School of Education	http://www.ius.edu/education/
School of Natural Sciences	http://www.ius.edu/naturalsciences/
School of Nursing	http://www.ius.edu/nursing/
School of Social Sciences	http://www.ius.edu/socialsciences/
Agriculture	http://www.coa.lsu.edu/
Art & Design	http://www.design.lsu.edu/
Business, E. J. Ourso	http://business.lsu.edu
Coast and Environment	http://www.sce.lsu.edu/
Engineering	http://www.eng.lsu.edu/
Human Sciences & Education	http://chse.lsu.edu
Humanities & Social Sciences	http://hss.lsu.edu/
Mass Communication, Manship School of	http://www.manship.lsu.edu/
Music & Dramatic Arts	http://www.cmda.lsu.edu/
Science	http://science.lsu.edu/
Veterinary Medicine	http://www1.vetmed.lsu.edu/svm/
Arts and Science, College of	http://www.cas.muohio.edu/
Business, Farmer School of	http://www.fsb.muohio.edu/
Creative Arts, College of	http://www.fna.muohio.edu/
Education, Health, and Society, College of	http://www.muohio.edu/eap
Engineering and Computing, College of	http://www.eas.muohio.edu/
Professional Studies and Applied Sciences, College of	http://www.regionals.muohio.edu/

School of Business and Economics	http://www.mtu.edu/business/
College of Engineering	http://www.mtu.edu/engineering/
School of Forest Resources and Environmental Science	http://www.mtu.edu/forest/
College of Sciences and Arts	http://www.mtu.edu/sciences-arts/
School of Technology	http://www.mtu.edu/technology/
College of Allied Health and Nursing	http://ahn.mnsu.edu/
College of Arts and Humanities	http://www.mnsu.edu/carts/
College of Business	http://cob.mnsu.edu/
College of Education	http://ed.mnsu.edu/
College of Science, Engineering and Technology	http://cset.mnsu.edu/
College of Social and Behavioral Sciences	http://sbs.mnsu.edu/
College of Arts, Media & Communication	http://www.mnstate.edu/camc/
College of Education & Human Services	http://www.mnstate.edu/cehs/
College of Humanities & Social Sciences	http://www.mnstate.edu/chss/
College of Science, Health & the Environment	http://www.mnstate.edu/cshe/

Table 30: Faculty List Construction Units

Faculty of Dentistry	http://www.mcgill.ca/dentistry/
Faculty of Law	http://www.mcgill.ca/law/
Desautels Faculty of Management	http://www.mcgill.ca/desautels/
Faculty of Religious Studies	http://www.mcgill.ca/religiousstudies/
Audiology and Speech Sciences, School of	http://www.audiospeech.ubc.ca/
Business, Sauder School of	http://www.sauder.ubc.ca/

Community and Regional Planning, School of	http://www.scarp.ubc.ca/
Dentistry, Faculty of	http://www.dentistry.ubc.ca/
Journalism, School of	http://www.journalism.ubc.ca/
Kinesiology, School of	http://www.kin.ubc.ca/
Nursing, School of	http://www.nursing.ubc.ca/
Population and Public Health, School of	http://www.spph.ubc.ca/
Pharmaceutical Sciences, Faculty of	http://www.pharmacy.ubc.ca/
Information, School of	http://www.ischool.berkeley.edu/
Journalism, Graduate School of	http://journalism.berkeley.edu/
Law, School of	http://www.law.berkeley.edu/
Public Policy, Richard & Rhoda Goldman School of	http://gspp.berkeley.edu/
Social Welfare, School of	http://socialwelfare.berkeley.edu/
Business, Graduate School of	http://www.gsb.stanford.edu/
Harvard Graduate School of Education	http://www.gse.harvard.edu/
Harvard Kennedy School	http://www.hks.harvard.edu/
Harvard Law School	http://www.law.harvard.edu/index.html
Divinity School	http://www.divinity.duke.edu/
Fuqua School of Business	http://www.fuqua.duke.edu/
School of Law	http://www.law.duke.edu/
The Betty Irene Moore School of Nursing	http://www.ucdmc.ucdavis.edu/nursing/
Graduate School of Management	http://www.gsm.ucdavis.edu/
School of Education	http://education.ucdavis.edu/
College of Education	http://www.education.uiowa.edu/
College of Law	http://www.law.uiowa.edu/
Agricultural Economics	http://agrecon.mcgill.ca
Animal Science	http://www.mcgill.ca/animal
Bioresource Engineering	http://www.mcgill.ca/bioeng
Farm Management & Technology Program	http://www.mcgill.ca/fmt
Food Science	http://www.mcgill.ca/foodscience
Institute of Parasitology	http://www.mcgill.ca/parasitology

McGill School of Environment	http://www.mcgill.ca/mse
Natural Resource Sciences	http://www.mcgill.ca/nrs
Plant Science	http://www.mcgill.ca/plant
School of Dietetics and Human Nutrition	http://www.mcgill.ca/dietetics
Department of Forest Resources Management	http://frm.forestry.ubc.ca
Department of Forest and Conservation Sciences	http://www.forestry.ubc.ca/departments/forest-sciences/
Department of Wood Science	http://wood.ubc.ca
Anesthesiology, Pharmacology & Therapeutics	http://www.apt.ubc.ca/
Biochemistry & Molecular Biology	http://www.biochem.ubc.ca/
Cellular & Physiological Sciences	http://www.cellphys.ubc.ca/
Dermatology & Skin Science	http://www.derm.ubc.ca/
Emergency Medicine	http://www.emergency.med.ubc.ca/
Family Practice	http://www.familymed.ubc.ca/
Composition	http://www.music.ubc.ca/divisions/composition.html
Ethnomusicology	http://www.music.ubc.ca/divisions/ethnomusicology.html
Musicology	http://www.music.ubc.ca/divisions/musicology.html
Music Theory	http://www.music.ubc.ca/divisions/theory.html
Botany	http://www.botany.ubc.ca
Chemistry	http://www.chem.ubc.ca
Computer Science	http://www.cs.ubc.ca
Earth, Ocean and Atmospheric Sciences	http://www.eos.ubc.ca
Mathematics	http://www.math.ubc.ca
Microbiology and Immunology	http://www.microbiology.ubc.ca
Physics and Astronomy	http://www.phas.ubc.ca
Statistics	http://www.stat.ubc.ca
Zoology	http://www.zoology.ubc.ca

Anthropology	http://www.chass.utoronto.ca/anthropology
Art	http://www.art.utoronto.ca
Astronomy & Astrophysics	http://www.astro.utoronto.ca
Cell & Systems Biology	http://www.csb.utoronto.ca
Chemistry	http://www.chem.utoronto.ca
Classics	http://www.chass.utoronto.ca/classics
Computer Science	http://www.cs.utoronto.ca
Anaesthesia	http://www.dentistry.utoronto.ca/departments/anaesthesia
Biomaterials	http://www.dentistry.utoronto.ca/departments/biomaterials
Dental Public Health	http://www.dentistry.utoronto.ca/departments/dental-public-health
Endodontics	http://www.dentistry.utoronto.ca/departments/endodontics
Oral Microbiology	http://www.dentistry.utoronto.ca/departments/oral-microbiology
Department of Geography	http://www.uoguelph.ca/geography
Department of Psychology	http://www.uoguelph.ca/psychology
Department of Political Science	http://www.uoguelph.ca/polisci
Department of Business	http://www.business.uoguelph.ca/
Department of Economics and Finance	http://www.uoguelph.ca/economics/
Department of Marketing and Consumer Studies	http://www.uoguelph.ca/mcs/
Bioengineering (BioE)	http://bioeng.berkeley.edu/
Civil & Environmental Engineering (CEE)	http://www.ce.berkeley.edu/index.php
Electrical Engineering & Computer Sciences (EECS)	http://www.eecs.berkeley.edu/
Industrial Engineering & Operations Research (IEOR)	http://www.ieor.berkeley.edu/
Materials Science & Engineering (MSE)	http://www.mse.berkeley.edu/
Mechanical Engineering (ME)	http://www.me.berkeley.edu/
Nuclear Engineering (NE)	http://www.nuc.berkeley.edu/

Animal Science	http://www.ansci.cornell.edu/index.html
Charles H. Dyson School of Applied Economics and Management	http://aem.cornell.edu/
Biological and Environmental Engineering	http://www.bee.cornell.edu/
Biological Statistics and Computational Biology	http://www.bscb.cornell.edu/
Communication	http://www.comm.cornell.edu/
Crop and Soil Sciences	http://css.cals.cornell.edu/
Development Sociology	http://devsoc.cals.cornell.edu/
Applied and Engineering Physics	http://www.aep.cornell.edu
Biomedical Engineering	http://www.bme.cornell.edu/
Civil and Environmental Engineering	http://www.cee.cornell.edu/
Earth and Atmospheric Sciences	http://www.eas.cornell.edu/
Accounting	http://www.krannert.purdue.edu/academics/Accounting/home.asp
Economics	http://www.krannert.purdue.edu/academics/Economics/home.asp
Finance	http://www.krannert.purdue.edu/academics/Finance/home.asp
Department of Civil & Environmental Engineering	http://cee.duke.edu/
Department of Electrical & Computer Engineering	http://www.ece.duke.edu/
Department of Mechanical Engineering & Materials Science	http://mems.duke.edu/
Art	http://studioart.arts.uci.edu/
Dance	http://dance.arts.uci.edu/
Drama	http://drama.arts.uci.edu/
Music	http://music.arts.uci.edu/
Computer Science Department	http://www.ics.uci.edu/computerscience
Informatics Department	http://www.ics.uci.edu/informatics/
Statistics Department	http://www.ics.uci.edu/statistics

Criminology, Law and Society	http://cls.soceco.uci.edu/
Psychology and Social Behavior	http://psb.soceco.uci.edu/
Planning, Policy and Design	http://ppd.soceco.uci.edu/
Cognitive Sciences	http://www.cogsci.uci.edu
Economics	http://www.economics.uci.edu
Logic and Philosophy of Science	http://www.lps.uci.edu
Political Science	http://www.polisci.uci.edu
Sociology	http://www.sociology.uci.edu
Bioengineering	http://www.bioen.utah.edu
Chemical Engineering	http://www.che.utah.edu
Civil & Environmental Engineering	http://www.civil.utah.edu/
Electrical & Computer Engineering	http://www.ece.utah.edu
Materials Science & Engineering	http://www.mse.utah.edu/
Mechanical Engineering	http://www.mech.utah.edu
School of Computing	http://www.cs.utah.edu/
Communication	http://communication.utah.edu/index.php
English	http://english.utah.edu/index.php
History	http://history.utah.edu/index.php
Languages & Literature	http://languages.utah.edu/index.php
Linguistics	http://linguistics.utah.edu/index.php
Philosophy	http://philosophy.utah.edu/index.php
Department of Biology	http://www.biology.utah.edu/
Department of Chemistry	http://www.chem.utah.edu/
Department of Mathematics	http://www.math.utah.edu/
Department of Physics & Astronomy	http://www.physics.utah.edu/
Anthropology	http://www.anthro.utah.edu/
Economics	http://www.econ.utah.edu/
Family and Consumer Studies	http://www.fcs.utah.edu/
Geography	http://www.geog.utah.edu/
Political Science	http://www.poli-sci.utah.edu/

Psychology	http://www.psych.utah.edu/
Sociology	http://www.soc.utah.edu/
Biomedical Engineering	http://www.bme.ncsu.edu/
Chemical and Biomolecular Engineering	http://www.che.ncsu.edu/
Civil, Construction, and Environmental Engineering	http://www.ce.ncsu.edu/
Computer Science	http://www.csc.ncsu.edu/
Electrical and Computer Engineering	http://www.ece.ncsu.edu/
Edward P. Fitts Industrial and Systems Engineering	http://www.ise.ncsu.edu/
Materials Science and Engineering	http://www.mse.ncsu.edu/

Table 32: Faculty List Testing Units

College of Nursing	http://www.nursing.uiowa.edu/
Information & Media Studies	http://www.fims.uwo.ca/index.htm
Law	http://www.law.uwo.ca/
Ivey Business School	http://www.ivey.uwo.ca/
Rady School of Management	http://rady.ucsd.edu/
School of International Relations and Pacific Studies	http://irps.ucsd.edu/
Scripps Institution of Oceanography	http://www.sio.ucsd.edu/
Skaggs School of Pharmacy & Pharmaceutical Sciences	http://pharmacy.ucsd.edu/index.shtml
McIntire School of Commerce	http://www.commerce.virginia.edu/
Darden School of Business	http://www.darden.virginia.edu/
Curry School of Education	http://curry.virginia.edu/
Business	http://merage.uci.edu/
Education	http://gse.uci.edu/
Nursing Science	http://www.nursing.uci.edu/
Pharmaceutical Sciences	http://pharmsci.uci.edu/

Journalism and Communications	http://www.jou.ufl.edu/
Law	http://www.law.ufl.edu/
Journalism	http://cronkite.asu.edu
Law	http://www.law.asu.edu/
Nursing and Health Innovation	http://nursingandhealth.asu.edu/
Sustainability	http://schoolofsustainability.asu.edu
Technology and Innovation	http://technology.poly.asu.edu/
Education	http://www.csus.edu/coe/
Business Administration	http://www.cba.csus.edu/
College of Business	http://www3.dbu.edu/business/
Elliott School of International Affairs	http://elliott.gwu.edu/
School of Nursing	http://nursing.gwu.edu/
School of Business	http://www.ius.edu/business/
School of Education	http://www.ius.edu/education/
Mass Communication, Manship School of	http://www.manship.lsu.edu/
The School of the Arts	http://sota.mcmaster.ca/
The Department of Classics	http://www.humanities.mcmaster.ca/~classics
The Department of Communication Studies & Multimedia	http://csmm.mcmaster.ca/
The Department of English & Cultural Studies	http://www.humanities.mcmaster.ca/~english
The Department of French	http://www.humanities.mcmaster.ca/~french
The Department of History	http://www.humanities.mcmaster.ca/~history
Classical Studies	http://www.uwo.ca/classics
English & Writing Studies	http://www.uwo.ca/arts/pages/english-writing.html
Film Studies	http://www.uwo.ca/film
French Studies	http://www.uwo.ca/french/
Linguistics	http://www.uwo.ca/linguistics/
Modern Languages	http://www.uwo.ca/modlang/
Philosophy	http://www.uwo.ca/philosophy

Applied Mathematics	http://www.uwo.ca/sci/departments/applied_math.html
Basic Medical Sciences	http://www.uwo.ca/sci/departments/medical_sciences.html
Biology	http://www.uwo.ca/sci/departments/biology.html
Chemistry	http://www.uwo.ca/sci/departments/chemistry.html
Computer Science	http://www.uwo.ca/sci/departments/computer_science.html
Earth Sciences	http://www.uwo.ca/sci/departments/earth_science.html
Mathematics	http://www.uwo.ca/sci/departments/mathematics.html
Physics & Astronomy	http://www.uwo.ca/sci/departments/physics_astronomy.html
School ofArchitecture	http://architecture.uwaterloo.ca/
Department of Chemical Engineering	http://uwaterloo.ca/chemical-engineering
Department of Electrical & Computer Engineering	http://ece.uwaterloo.ca/Home/
Department of Management Sciences	http://uwaterloo.ca/management-sciences
Department of Systems Design Engineering	http://www.syde.uwaterloo.ca/
Applied Mathematics Department	http://math.uwaterloo.ca/applied-mathematics/
David R. Cheriton School of Computer Science	https://cs.uwaterloo.ca
Pure Mathematics Department	http://math.uwaterloo.ca/pure-mathematics/
Department of English Language and Literature	http://www.carleton.ca/english/
Department of French	http://www.carleton.ca/french/
Department of Geography and Environmental Studies	http://www.carleton.ca/geography/
Department of History	http://www.carleton.ca/history/
Department of Philosophy	http://www.carleton.ca/philosophy/
Department of Psychology	http://www.carleton.ca/psychology/
Department of Sociology and Anthropology	http://www.carleton.ca/socanth/

Department of Civil and Environmental Engineering	http://www1.carleton.ca/cee/
Department of Electronics	http://www.doe.carleton.ca
Department of Mechanical and Aerospace Engineering	http://www1.carleton.ca/mae/
Department of Systems and Computer Engineering	http://sce.carleton.ca
Department of Biology	http://www.carleton.ca/biology/
Department of Chemistry	http://www.carleton.ca/chem/
Department of Earth Sciences	http://www.earthsci.carleton.ca/
Department of Neuroscience	http://www.carleton.ca/neuroscience/
Department of Physics	http://www.physics.carleton.ca/
Institute of Biochemistry	http://www.carleton.ca/biochem/
Institute of Environmental Science	http://www.carleton.ca/envirosci/
Integrated Science Institute	http://www.carleton.ca/isi/
School of Computer Science	http://www.scs.carleton.ca/
Anesthesiology	http://anes-som.ucsd.edu/
Cellular and Molecular Medicine	http://cmm.ucsd.edu/
Emergency Medicine	http://emergencymed.ucsd.edu
Family and Preventive Medicine	http://famprevmed.ucsd.edu/
Medicine	http://med.ucsd.edu/
Neurosciences	http://neurosciences.ucsd.edu/
Ophthalmology	http://eyesite.ucsd.edu/
Orthopaedic Surgery	http://ortho.ucsd.edu
Pathology	http://pathology.ucsd.edu
Africana Studies	https://africana.sas.upenn.edu/department
Anthropology	http://www.sas.upenn.edu/anthro/
Biology	http://www.bio.upenn.edu/
Chemistry	http://www.sas.upenn.edu/chem/
Classical Studies	http://www.classics.upenn.edu/index.html
Criminology	http://www.crim.upenn.edu/
Earth and Environmental Science	http://www.sas.upenn.edu/earth/

East Asian Languages & Civilizations	http://www.sas.upenn.edu/ealc/
Economics	http://economics.sas.upenn.edu/
English	http://www.english.upenn.edu/
Biomedical	http://bme.virginia.edu/
Chemical	http://www.che.virginia.edu/
Civil & Environmental	http://ce.virginia.edu/
Computer Science	http://www.cs.virginia.edu/
Electrical & Computer	http://www.ee.virginia.edu/
Engineering and Society	http://www.eands.virginia.edu/eands/
Materials Science & Engineering	http://www.virginia.edu/ms/
Mechanical & Aerospace	http://www.mae.virginia.edu/
Systems & Information	http://www.sys.virginia.edu/
Biomedical Engineering	http://www.eng.uci.edu/dept/bme
Chemical Engineering & Materials Science	http://www.eng.uci.edu/dept/chems
Civil & Environmental Engineering	http://www.eng.uci.edu/dept/cee
Electrical Engineering & Computer Science	http://www.eng.uci.edu/dept/eecs
Mechanical & Aerospace Engineering	http://mae.eng.uci.edu/
Anthropology Department	http://www.anthro.ufl.edu/
Astronomy Department	http://www.astro.ufl.edu/
Biology	http://www.biology.ufl.edu/
Chemistry Department	http://www.chem.ufl.edu/
Classics Department	http://www.classics.ufl.edu/
English Department	http://www.english.ufl.edu/
Geography Department	http://www.geog.ufl.edu/
Geological Sciences, Department of	http://www.geology.ufl.edu/
History Department	http://www.history.ufl.edu/
Infectious Diseases & Pathology	http://idp.vetmed.ufl.edu/
Small Animal Clinical Sciences	http://sacs.vetmed.ufl.edu/
Large Animal Clinical Sciences	http://lacs.vetmed.ufl.edu/
Physiological Sciences	http://physio.vetmed.ufl.edu
Arts, Media + Engineering	http://ame.asu.edu/

Dance	http://dance.asu.edu/
Design	http://design.asu.edu/
Film, Dance and Theatre	http://theatrefilm.asu.edu/
Applied Mathematics & Statistics	http://ams.soe.ucsc.edu/
Biomolecular Engineering	http://bme.soe.ucsc.edu/
Computer Engineering	http://ce.soe.ucsc.edu/
Computer Science	http://cs.soe.ucsc.edu/
Electrical Engineering	http://ee.soe.ucsc.edu/
Technology & Information Management	http://tim.soe.ucsc.edu/
Civil Engineering	http://www.ecs.csus.edu/ce
Computer Engineering	http://www.ecs.csus.edu/cpe
Computer Science	http://www.ecs.csus.edu/csc
Construction Management	http://www.ecs.csus.edu/cm
Electrical & Electronic Engineering	http://www.ecs.csus.edu/eee
Mechanical Engineering	http://www.ecs.csus.edu/me
Anthropology	http://www.csus.edu/anth/
Asian Studies	http://www.csus.edu/ASIA/
Economics	http://www.csus.edu/econ/
Environmental Studies	http://www.csus.edu/envs/
Ethnic Studies	http://www.csus.edu/ethn/
Family & Consumer Sciences	http://www.csus.edu/facs/
Department of Teacher Education	http://www.emich.edu/coe/ted/index.php
Department of Special Education	http://www.emich.edu/coe/sped/index.php
Department of Leadership & Counseling	http://www.emich.edu/coe/lc/index.php

Table 34: Page Retrieval Keywords

File extensions for crawler filter	css, js, bmp, gif, jpe?g, png, tiff?, mid, mp2, mp3, mp4, wav, avi, mov, mpeg, ram, m4v, pdf, doc, docx, xls, xlsx, ppt, pptx, xml, rm, smil, wmv, swf, wma, zip, rar, gz

Keywords for page retrieval heuristics	academics, academic divisions, academic areas, academic units, faculties, colleges, divisions, schools, departments, institutes, programs, department list, about the college, about the school, about the faculty, fields, majors, faculty, directory, profile, people, professors, members, staff, committee

List of References

[1] "Semantic web." http://www.w3.org/standards/semanticweb/. [Online; accessed 13-September-2013].

[2] M. Gilula. "Structured search: From keywords to key-objects." http://strictsearch.com/send-doc.html. [Online; accessed 8-September-2013] (2012).

[3] R. Grishman. "Information extraction: Techniques and challenges." In "International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology," SCIE '97, pages 10–27. Springer-Verlag, London, UK, UK. ISBN 3-540-63438-X (1997).

[4] J. Tang, M. Hong, D. Zhang, B. Liang, and J. Li. "Information extraction: Methodologies and applications."

[5] W. W. W. Consortium. "Document object model (dom)." http://www.w3.org/DOM/. [Online; accessed 4-April-2013].

[6] J. Hedley. "jsoup: Java html parser." http://jsoup.org/. [Online; accessed 4-April-2013].

[7] w3schools. "The css box model." http://www.w3schools.com/css/css_boxmodel.asp. [Online; accessed 4-April-2013].

[8] R. Burget. "Cssbox - java html rendering engine." http://cssbox.sourceforge.net/. [Online; accessed 4-April-2013].

[9] C. Castillo. "Effective web crawling." *SIGIR Forum* **39**(1), 55–56. ISSN 0163-5840 (2005).

[10] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition. ISBN 0408709294 (1979).

[11] N. Kushmerick, D. S. Weld, and R. Doorenbos. "Wrapper induction for information extraction." (1997).

[12] N. Kushmerick. "Wrapper induction: efficiency and expressiveness." *Artif. Intell.* **118**(1-2), 15–68. ISSN 0004-3702 (2000).

[13] I. Muslea, S. Minton, and C. A. Knoblock. "Hierarchical wrapper induction for semistructured information sources." *Autonomous Agents and Multi-Agent Systems* **4**(1-2), 93–114. ISSN 1387-2532 (2001).

[14] C.-N. Hsu and M.-T. Dung. "Generating finite-state transducers for semi-structured data extraction from the web." *Inf. Syst.* **23**(9), 521–538. ISSN 0306-4379 (1998).

[15] S. Soderland. "Learning information extraction rules for semi-structured and free text." *Mach. Learn.* **34**(1-3), 233–272. ISSN 0885-6125 (1999).

[16] D. Freitag. "Machine learning for information extraction in informal domains." *Mach. Learn.* **39**(2-3), 169–202. ISSN 0885-6125 (2000).

[17] W. Liu, X. Meng, and W. Meng. "Vide: A vision-based approach for deep web data extraction." *IEEE Trans. on Knowl. and Data Eng.* **22**(3), 447–460. ISSN 1041-4347 (2010).

[18] B. Liu, R. Grossman, and Y. Zhai. "Mining data records in web pages." In "Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining," KDD '03, pages 601–606. ACM, New York, NY, USA. ISBN 1-58113-737-0 (2003).

[19] Y. Zhai and B. Liu. "Web data extraction based on partial tree alignment." In "Proceedings of the 14th international conference on World Wide Web," WWW '05, pages 76–85. ACM, New York, NY, USA. ISBN 1-59593-046-9 (2005).

[20] B. Liu and Y. Zhai. "Net - a system for extracting web data from flat and nested data records." In "Proceedings of 6th International Conference on Web Information Systems Engineering," WISE '05, pages 487–495 (2005).

[21] V. Crescenzi, G. Mecca, P. Merialdo, U. Roma, T. Universit, B. Universit, and R. Tre. "Roadrunner: Towards automatic data extraction from large web sites." pages 109–118 (2001).

[22] C.-H. Chang and S.-C. Lui. "Iepad: information extraction based on pattern discovery." In "Proceedings of the 10th international conference on World Wide Web," WWW '01, pages 681–688. ACM, New York, NY, USA. ISBN 1-58113-348-0 (2001).

[23] A. Arasu and H. Garcia-Molina. "Extracting structured data from web pages." In "Proceedings of the 2003 ACM SIGMOD international conference on Management of data," SIGMOD '03, pages 337–348. ACM, New York, NY, USA. ISBN 1-58113-634-X (2003).

[24] J. Wang and F. H. Lochovsky. "Data extraction and label assignment for web databases." In "Proceedings of the 12th international conference on World Wide Web," WWW '03, pages 187–196. ACM, New York, NY, USA. ISBN 1-58113-680-3 (2003).

[25] V. Vapnik. *Statistical learning theory*. Wiley. ISBN 978-0-471-03003-4 (1998).

[26] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. "A maximum entropy approach to natural language processing." *COMPUTATIONAL LINGUISTICS* **22**, 39–71 (1996).

[27] Y. Freund and R. E. Schapire. "A short introduction to boosting." In "In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence," pages 1401–1406. Morgan Kaufmann (1999).

[28] M. Collins. "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms." In "Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10," EMNLP '02, pages 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA (2002).

[29] Z. Ghahramani, M. I. Jordan, and P. Smyth. "Factorial hidden markov models." In "Machine Learning," MIT Press (1997).

[30] A. McCallum, D. Freitag, and F. C. N. Pereira. "Maximum entropy markov models for information extraction and segmentation." In "Proceedings of the Seventeenth International Conference on Machine Learning," ICML '00, pages 591–598. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1-55860-707-2 (2000).

[31] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." In "Proceedings

of the Eighteenth International Conference on Machine Learning," ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1-55860-778-1 (2001).

[32] A. Janevski. *UniversityIE: Information Extraction From University Web Pages*. Master's thesis, University of Kentucky (2000).

[33] A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. "A brief survey of web data extraction tools." *SIGMOD Rec.* **31**(2), 84–93. ISSN 0163-5808 (2002).

[34] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. "A survey of web information extraction systems." *IEEE Trans. on Knowl. and Data Eng.* **18**(10), 1411–1428. ISSN 1041-4347 (2006).

[35] S. Sarawagi. "Information extraction." *Found. Trends databases* **1**(3), 261–377. ISSN 1931-7883 (2008).

[36] E. Ferrara, P. D. Meo, G. Fiumara, and R. Baumgartner. "Web data extraction, applications and techniques: A survey." *CoRR* **abs/1207.0246** (2012).

[37] W3Schools. "Introduction to web services." `http://www.w3schools.com/webservices/ws_intro.asp`. [Online; accessed 7-October-2013] (2013).

[38] . Digits. "Web extractor." `http://www.30digits.com/web-extractor-2.htm`. [Online; accessed 7-October-2013] (2013).

[39] Diffbot. "Diffbot." `http://www.diffbot.com/`. [Online; accessed 7-October-2013] (2013).

[40] I. AlchemyAPI. "Introduction to web services." `http://www.alchemyapi.com/`. [Online; accessed 7-October-2013] (2013).

[41] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. "Towards domain-independent information extraction from web tables." In "Proceedings of the 16th international conference on World Wide Web," WWW '07, pages 71–80. ACM, New York, NY, USA. ISBN 978-1-59593-654-7 (2007).

[42] S. Rajasurya, T. Muralidharan, S. Devi, and S. Swamynathan. "Semantic information retrieval using ontology in university domain." *CoRR* **abs/1207.5745** (2012).

[43] T. Zhou, C. Sun, L. Lin, and B. Liu. "An information extraction system for heterogeneous web source." In "ICMLC'10," pages 3287–3292 (2010).

[44] Y. Xiong, P. Luo, Y. Zhao, F. Lin, S. Feng, B. Zhou, and L. Zheng. "Ofcourse: web content discovery, classification and information extraction for online course materials." In "Proceedings of the 18th ACM conference on Information and knowledge management," CIKM '09, pages 2077–2078. ACM, New York, NY, USA. ISBN 978-1-60558-512-3 (2009).

[45] T. Weninger, M. Danilevsky, F. Fumarola, J. Hailpern, J. Han, T. J. Johnston, S. Kallumadi, H. Kim, Z. Li, D. McCloskey, Y. Sun, N. E. TeGrotenhuis, C. Wang, and X. Yu. "Winacs: construction and analysis of web-based computer science information networks." In "Proceedings of the 2011 ACM SIGMOD International Conference on Management of data," SIGMOD '11, pages 1255–1258. ACM, New York, NY, USA. ISBN 978-1-4503-0661-4 (2011).

[46] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. "Arnetminer: extraction and mining of academic social networks." In "Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining," KDD '08, pages 990–998. ACM, New York, NY, USA. ISBN 978-1-60558-193-4 (2008).

[47] A. Ameen, K. U. R. Khan, and B. Rani. "Construction of university ontology." In "Information and Communication Technologies (WICT), 2012 World Congress on," WICT '12, pages 39–44. ACM. ISBN 978-1-4673-4806-5 (2012).

[48] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. "Extracting content structure for web pages based on visual representation." In "Proceedings of the 5th Asia-Pacific web conference on Web technologies and applications," APWeb'03, pages 406–417. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-02354-2 (2003).

[49] L. Yao, J. Tang, and J. Li. "A unified approach to researcher profiling." In "Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence," WI '07, pages 359–366. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-3026-5 (2007).

[50] S. Dewanto. "Structured data extractor." `http://seagatesoft.blogspot.com/2012/05/structured-data-extractor.html`. [Online; accessed 2-September-2013] (2012).

[51] U. S. C. Bureau. "Genealogy data: Frequently occurring surnames from census 2000." http://www.census.gov/genealogy/www/data/2000surnames/index.html. [Online; accessed 10-April-2013].

[52] "Opencv." http://opencv.org/. [Online; accessed 10-April-2013].

[53] M. Liu and J. Hu. "Information networking model." In "Proceedings of the 28th International Conference on Conceptual Modeling," ER '09, pages 131–144. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-642-04839-5 (2009).