Discovering Web Structure with Multiple Experts in a Clustering Framework

Bora Cenk Gazen

CMU-CS-08-154

December 2008

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jaime Carbonell, Carnegie Mellon University, Chair
William Cohen, Carnegie Mellon University
John Lafferty, Carnegie Mellon University
Steven Minton, Fetch Technologies

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Structure discovery, heterogeneous experts, hypothesis language, confidence scores, clustering, unsupervised data extraction, world wide web, record linkage

Abstract

The world wide web contains vast amounts of data, but only a small portion of it is accessible in an operational form by machines. The rest of this vast collection is behind a presentation layer that renders web pages in a human-friendly form but also hampers machine-processing of data. The task of converting web data into operational form is the task of *data extraction*. Current approaches to data extraction from the web either require human-effort to guide supervised learning algorithms or are customized to extract a narrow range of data types in specific domains. We focus on the broader problem of discovering the underlying structure of any database-generated web site. Our approach automatically discovers relational data that is hidden behind these web sites by combining experts that identify the relationship between surface structure and the underlying structure.

Our approach is to have a set of *software experts* that analyze a web site's pages. Each of these experts is specialized to recognize a particular type of structure. These experts discover similarities between data items within the context of the particular types of structure they analyze and output their discoveries as hypotheses in a common hypothesis language. We find the most likely clustering of data using a probabilistic framework in which the hypotheses provide the evidence. From the clusters, the relational form of the data is derived.

We develop two frameworks following the principles of our approach. The first framework introduces a common hypothesis language in which heterogeneous experts express their discoveries. The second framework extends the common language to allow experts to assign confidence scores to their hypotheses.

We experiment in the web domain by comparing the output of our approach to the data extracted by a supervised wrapper-induction system and validated manually. Our results show that our approach performs well in the data extraction task on a variety of web sites.

Our approach is applicable to other structure discovery problems as well. We demonstrate this by successfully applying our approach in the record deduplication domain.

Acknowledgments

I have been fortunate to have great teachers. Without their guidance, I would have not made it this far. I would especially like to thank Ari Requicha, who introduced me to research and academia, Craig Knoblock, who encouraged me to pursue my academic ambitions, Jaime Carbonell, who provided me inspiration with his amazingly deep and broad knowledge and intellect, and Steve Minton, who supported me with his continuous guidance and encouragement. Thanks to William Cohen and John Lafferty for their patient and meticulous supervision.

I also want to thank everyone at Fetch for supporting me during the last 6 years. In particular, I want to thank Evan Gamble for helping me with the thesis project and for the great discussions on science, and Greg Barish and Sofus Macskassy for sharing their PhD experiences and encouraging me to keep working on mine.

Also, many thanks to Steven Spitz. Having him to talk to has kept me sane through the many years in graduate school.

Finally, I would like to thank my parents and my wife, Pui. My dad has always reminded me that I could do better when I thought I had done my best and celebrated my success when I felt I could have done better. And Pui, without your love and support, I would have quit long time ago. Thank you for being my partner in life. We accomplished all this together.

Contents

1 Introduction **1**

1.1 Overview . . . 1
1.1.1 Goal . . . 1
1.1.2 Current Approaches . . . 2
1.1.3 Challenges . . . 3
1.1.4 Contribution . . . 4
1.2 Problem Description . . . 5
1.2.1 Site extraction . . . 5
1.2.2 Inverse Problem - Site Generation . . . 7
1.3 Thesis Statement . . . 8
1.4 Roadmap . . . 9

2 Approach - Principles **11**

2.1 Motivation . . . 11
2.2 Structure Discovery . . . 13
2.3 Tackling Structure Discovery Problems . . . 14
2.3.1 Multiple Heterogeneous Experts . . . 15
2.3.2 Combining Heterogeneous Experts . . . 16
2.3.3 Experts vs. Complex Models . . . 17
2.4 Summary . . . 18

3 CHEX: Clustering with Heterogeneous Experts **19**

3.1 Overview . . . 19
3.2 Expert Framework . . . 20
3.3 Hypothesis Language . . . 22
3.4 Probabilistic Model . . . 23
3.5 Clustering . . . 26
3.6 Finding Data Clusters . . . 28
3.7 Web Experts . . . 30
3.8 Clusters to Relational form . . . 36
3.9 Summary . . . 41

4 Results - CHEX **43**
4.1 Evaluation Methodology . . . 43
4.1.1 Dataset . . . 43
4.1.2 Precision/Recall/F1 on Matching Clusters . . . 44
4.2 Results . . . 45
4.3 Observations . . . 52

5 CONFHEX: Clustering with Confidence Scores from Heterogeneous Experts **53**
5.1 Overview . . . 53
5.2 Bayesian Network . . . 54
5.2.1 Structure . . . 54
5.2.2 Parameters . . . 56
5.2.3 Confidence Scores . . . 57
5.2.4 Belief Propagation . . . 59
5.3 Combining Experts . . . 60
5.4 Web Domain . . . 63
5.4.1 Overview . . . 63
5.4.2 Page Clustering . . . 63
5.4.3 Data Clustering . . . 65
5.5 Summary . . . 69

6 Results - CONFHEX **71**
6.1 Goals . . . 71
6.2 CONFHEX on the Web Dataset . . . 71
6.2.1 Page Clustering . . . 72
6.2.2 Data Clustering . . . 72
6.3 Comparing CONFHEX to AgentBuilder . . . 78
6.4 Comparing CONFHEX to RoadRunner . . . 79
6.5 Comparing CONFHEX to CHEX . . . 81
6.6 Applications . . . 82
6.6.1 News Extraction . . . 82

7 Record Deduplication **87**
7.1 Overview . . . 87
7.1.1 Citation Experts . . . 90
7.1.2 Venue experts . . . 91
7.2 Experiments . . . 91
7.2.1 Results . . . 91
7.3 Summary . . . 92

8 Related Work **93**
8.1 Data Extraction from the Web . . . 93
8.1.1 Wrappers . . . 93
8.1.2 Table Extraction . . . 95

8.1.3 Grammar Induction . . . 96
8.1.4 Unsupervised Extraction . . . 97
8.2 Relational Model Learning. . . . 98
8.2.1 Data Mining in Graphs. . . . 99
8.3 Clustering . . . 100
8.3.1 Coclustering . . . 101
8.3.2 Clustering with Constraints . . . 101
8.3.3 Ensemble Clustering . . . 102
8.4 Multi-Expert Approaches . . . 103
8.4.1 GRAVA - Self-Adaptive Architecture . . . 104
8.4.2 Proverb - A Crossword Solver . . . 105
8.4.3 Poirot - Integrated Learning . . . 106

9 Conclusions and Future Work **107**
9.1 Summary . . . 107
9.2 Contributions . . . 108
9.3 Future work . . . 109
9.3.1 Theoretical Framework . . . 109
9.3.2 Representation of Underlying Structure . . . 110
9.3.3 Iterative Interpretation . . . 110
9.3.4 Other Domains . . . 110

Bibliography **113**

List of Figures

1.1 Extracted Data in Relational Form — The goal of our approach is to recover the relational structure of web data. Here information about a particular book has been extracted and put into a relational table. 6

1.2 Relational Model of a Web Site — Two relational tables store the data presented on this web site. Each row of the States table corresponds to a page that displays information about a particular state. Similarly, each row of the CityWeather table corresponds to a page that displays information about a particular city. 8

3.1 Clusters of Pages and Tokens — Pages and tokens are clustered at the same time while maintaining the constraint that tokens from two pages can be in the same token cluster only if the pages are in the same page cluster. . . 29

3.2 Alignment of Fields on a Web Page — On web pages that display tabular data, the alignment of data items as seen on the screen usually reflects the underlying relational structure of the data. Columns of the relational table are aligned parallel to a vertical axis and rows are parallel to a horizontal axis. 36

3.3 Relational Structure of Data — References to values in other tables have been replaced with the actual values. For example, the second column in the first row refers to values stored in another table. These values, [s2,s3],[s4,s5],[s6,s7], are shown as an inner table within a cell of the outer table. 37

3.4 Merge Steps to Compose Tables from Clusters — We merge clusters, each of which corresponds to a column, in a bottom-up fashion to form tables. 39

4.1 Target and Extracted Clusters – To evaluate the accuracy of clustering, we compare the values in each target cluster with the values in each one of the extracted clusters. For each target cluster, we pick the extracted cluster with the highest recall score. Thus, cluster A is evaluated against cluster 1, cluster B against cluster 4 and cluster C also against cluster 4. Note that because each value is placed in exactly one extracted cluster, it is not possible to achieve recall and precision scores of 1.0 simply by generating all possible clusters of values. 46

5.1 Bayesian Network for Clustering — Evidence as collected by the experts enter the network at the leaf nodes and determines the probability of a pair of samples being in the same cluster. This is then used to find the most likely assignment to the root node C which ranges over all possible clusterings of the samples. 55
5.2 Sample Probability Tables — The probability table for the root node C reflects that all clusterings are equally likely, the probability table for L_{ab} reflects the deterministic relation between a particular clustering and the existence of a link between samples a and b, and the probability table for E_{1ab} reflects the confidence of expert E_1 on the hypothesis that a and b are in the same cluster. 56
5.3 Distribution of Distance Scores for an Expert — For this particular expert, the distribution of distance scores is skewed towards small values for pairs that are in the same cluster and towards larger values for those that are in separate clusters. 58
5.4 Sample HTML as Displayed in a Browser — Only text nodes, such as "Feels Like:" and "SW 14mph" are visible on screen. 65

6.1 Data Clustering Results — Histogram of precision, recall and F1 scores in bins of width 0.1. 75
6.2 Snippet from Sample Web Page — A name-value list such as the one on this page can be represented as multiple rows in a name-value table or as a single row where the names correspond to the columns of the table. . . . 77
6.3 Levels of Effort — We compare the levels of effort required to extract data from the web using three different approaches: AgentBuilder (a supervised rule induction system), RoadRunner (an unsupervised wrapper induction system), and CONFHEX (an unsupervised site-extraction system) 81

7.1 Record Linkage Results — Tacking the citation deduplication problem with knowledge-rich experts gives results comparable to those obtained with a more sophisticated and computationally-expensive framework. . . 92

List of Tables

4.1 Summary of Results — We ran experiments in three domains: Journals, E-Commerce Sites, and Job-Listings. We report precision, recall and F1 scores by comparison to data in fields extracted by wrappers induced by a supervised learning system. The output of CHEX includes these fields as well as all other data found on each site. 47

4.2 Extraction from E-Commerce Sites — All but one field is extracted with high accuracy across a number of sites. 48

4.3 Extraction from Electronic Journals — CHEX extracts almost all fields with 100% accuracy. 50

4.4 Extraction from Sites with Job Listings — On this domain, we evaluated CHEX on a larger dataset. 51

6.1 Clustering Web Pages — The goal of this experiment was to show that combining multiple experts leads to better results overall than using individual experts. The scores are pairwise-F1 scores as percentages. Bold font indicates the best score of each row. In the last row, we report p-values obtained via the paired two-tail Student's t-test. The p-value is the probability of observing the reported difference in the F1-scores with the assumption that there is no change in the performance of the system. Thus, lower p-values indicate that the results are unlikely to be coincidental. . . 73

6.2 Comparing CHEX and CONFHEX — We simulated CHEX using a modified version of CONFHEX to compare the two systems. CONFHEX achieves better precision scores than CHEX. 85

Chapter 1

Introduction

1.1 Overview

1.1.1 Goal

The web contains vast amounts of data, estimated to be in the order of tens of billions of pages[36] or even more when the hidden web, pages that are not accessible by simple links, is included[38]. Unfortunately, machines can only use a very small portion of this large data. This is because a human-browsable web requires a thick presentation layer and this layer makes data and its rich structure obscure for machines.

Consider the problem of populating an ontology with product information. There are many web sites that sell products online and provide detailed information (pricing, specifications, availability, etc.) about each product. Let's consider the problem of extracting all the product information from a single site to populate part of an ontology. Even within a single site the product data is spread across many pages, sometimes by automated mapping from a database, whose structure we wish to recover, and sometimes by manual embedding.

The sub-problem of collecting pages that contain relevant data is relatively easy to solve by spidering all the pages on the site, so we will focus on the problem of extracting product information. Product information can be represented as records with fields such as product name, product description, price, and specifications. If the specifications vary over different groups of products, they may be represented as name-value pairs in a separate set of records related to the parent product record. Example specifications are weight, dimensions, input power, number of pages, etc. For each product, it might also be useful

to extract associated reviews. Like the specifications, reviews may be represented as a related set of records where these records would have fields such as reviewer email, date the review was posted, and the text of the review. Our goal is to find all such information on a given site and transform it into a form that can be as easily used as if the data was stored in a relational database.

1.1.2 Current Approaches

One approach to extracting data from the web is to use wrappers[48]. A wrapper is a function that maps the text of the web page into the value of a particular field, which is normally a substring of the web page. Wrappers can be hand-written or induced by machine learning algorithms. A common tool for hand-writing wrappers is *regular expressions*[68]. A regular expression is a pattern that describes a collection of strings. For example, the regular expression "ab*a" describes all strings that start and end with the symbol "a" and have an arbitrary number of "b"s in between. *Applying* a regular expression to a string checks whether the string is within the set of strings described by the regular expression. A regular expression *matches* a string if the string is within the set of strings described by the regular expression. A wrapper can be constructed from a regular expression by applying the regular expression to all substrings of a web page. The result of extraction is then a substring that matches the regular expression. To make the extraction more predictable, substrings are typically ordered from left-to-right and longest-to-shortest. As a concrete example, take the regular expression "Price: $[0-9][0-9]*.[0-9][0-9]". This will match strings like the following: "Price: $1.79", "Price: $24.00", and "Price: $199.99", but will not match "Shipping: $2.00" or "Price: N/A". A wrapper constructed out of this regular expression will extract the first substring that contains the price information. For example, when applied to the input string "Availability: In Stock. Price: $1.79", the wrapper will extract "Price: $1.79" as this is the longest left-most substring that matches the regular expression.

Writing regular expressions by hand is a tedious task as it is difficult to gauge whether the pattern is too specific and too general without trying it out a number of sample pages. Alternatively, machine learning techniques[46] can be used to induce regular expression-like patterns from positive and negative training samples. With machine learning, a wrapper can be built simply by providing sufficient labeled training samples.

The main problem with wrappers, whether they are hand-built or induced by machine

learning, is that they are specific to a particular site or a data field or, more commonly, to both. Thus, extracting data from a site always requires some overhead whether it is in writing the wrappers by hand or providing training data to the wrapper induction algorithm. This overhead becomes especially significant when the problem involves extracting data from a large number of sites.

An entirely different approach to utilizing web data is building indices of web pages that can then be quickly searched to find the most relevant pages to a query. There are many web services, such as Google search, that provide this kind of access to web pages. These services, also called search engines, have a shallow understanding of web pages and build indices using simple models, such as a *bag-of-words* model, where the index keeps track of the number of occurrences of individual words on each page[65]. This kind of approach provides access to all pages that are relevant to a query as long the query can be formulated within the constraints of the search engine's model. Since a search engine has a shallow understanding of web pages, the range of queries and responses is narrow. For example, with a search engine one can find all web pages that refer to a particular product, especially if the product has a unique name or other identifier, because presumably any page that refers to the product will refer to it by its unique name. However, no simple indexing model can be sufficient when queries involve relationships between data fields on pages. For example, if the product query is extended to include only pages on which the price is below some threshold, a deeper understanding is necessary where at least the price displayed on the page is extracted and associated correctly with the product.

1.1.3 Challenges

The problem of data extraction from the web is a structure discovery problem, where the goal is to discover the structure of data from clues gathered from its representation. It is similar to other structure discovery problems such as speech recognition or vision in that the underlying structure of the data, such as text or a 3D scene, partially surfaces in its representation, such as a sound signal from a microphone or 2D image captured by a camera. On the web, the relational form of the data is obscured as its transformed into a human-browsable web site.

For example, on a given site, pages that contain the same type of information often have a similar layout on the screen. However, it is non-trivial to determine that two pages contain the same type of information only by looking at their layout. This is because

among all possible layout features only some will be related to the content of the pages and without additional structure it is impossible to choose the right ones. Similarly, on a given site, values of the same field will be formatted in the same manner, but again this structure on its own is not sufficient to reliably determine which value belongs to which field. For example, even though publishing dates are always in the form of "January 20, 2007" on a particular site, other types of dates on a page, such as the date a review is written, can easily have the same format. Thus, relying on the date format on such a site is not sufficient to correctly extract publishing date.

The challenge in web data extraction is to make clever use of heterogeneous types of structure as it surfaces in multiple forms.

1.1.4 Contribution

As we described earlier, current approaches to web data extraction focus on a shallow understanding (e.g., indexing of individual words) or have narrow scope (e.g., extraction from a particular site, extraction of person names). We focus on the broader problem of discovering the underlying structure of all data on any site. Our approach automatically discovers all relational data that is hidden behind many types of web sites by combining experts that identify the relationship between surface structure and the underlying structure. Web sites that our approach is well suited for are those that are or can be generated from relational data, such as retail sites, news sites, blogs, travel sites, job listing sites, weather sites, review sites, wikis, directories and so on. Most web sites fall into this category but there are some that don't such as web sites of an individual persons or hand-written sites with little or no structure.

We focus on the data extraction problem and not the spidering task, so we assume we are given a set of pages obtained by spidering a web site. The new approach works as follows: First, a set of software experts analyze the given set of pages. We define an *expert* as an algorithm that generates hypotheses about the solution to the extraction problem. Each of these experts concentrates on a particular type of structure. For example, we have experts that can induce common types of grammars for web pages, experts that analyze visual layout information, experts that can parse particular data types such as dates, experts that look for patterns in URLs, and so on. These experts discover similarities between text tokens based on the particular types of structure they analyze and output their discoveries as hypotheses in a common hypothesis language. Next, we find the most likely

clustering of the tokens using a probabilistic framework in which the hypotheses provide the evidence. Each cluster represents one particular field of the data. Thus, the clustering follows the underlying relational structure of the data closely and as a final post-processing step, we generate the relational form of the data from the most-likely clustering.

1.2 Problem Description

1.2.1 Site extraction

Given a set of *pages* from a *web site* and the *links* between the pages, the problem is to find a set of *relational tables* that best represent the data that is available on the pages. Next, we introduce some terminology and define the problem more formally.

The *web* is a collection of documents, or pages, linked to each other via *hyper-links*, which we will simply call *links*. A link on a document is defined by referring to the *URL* of another document. URLs are identifiers for web pages. A link can be fully or partially specified. When it is partially specified, the user provides input on a *web form*, which is a collection of user interface elements for selecting or typing text. The user input is then used to completely determine the link.

In addition to linking documents to one another, URLs are also useful for associating a page to its underlying data. This is done by including identifiers for the underlying data elements within the URL, e.g., as in "http://weather/current?zip=15213". Content on web pages is annotated with *HTML* which allows a web page developer to organize the content in a hierarchical fashion and assign formatting and semantic categories to the elements in the hierarchy. When HTML documents are parsed, they are often represented in the standard *Document Object Model*, often abbreviated as *DOM*.

In this work, a *web site* is a collection of pages that provide a consistent view of a data collection[1]. All web pages in a web site should be reachable by *spidering*, which is the process of automatically navigating through a web site and collecting all its pages. Automatically navigating through a web site involves extracting links on each page and retrieving the pages pointed by these links. Doing this for fully-specified links is straightforward. However, navigating through web forms is difficult in general because web forms need to be filled in with appropriate input before they can be spidered. In this dissertation,

[1]It is possible that a site contains multiple data collections. Our approach can then be applied to subsets of pages that correspond to individual collections

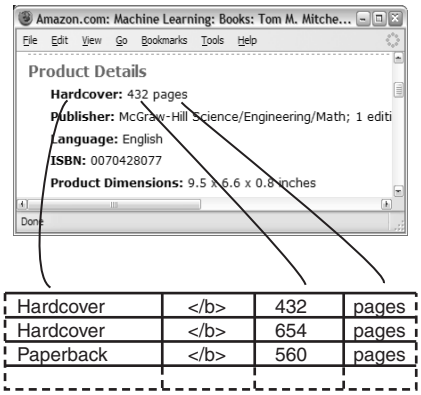


Figure 1.1: Extracted Data in Relational Form — The goal of our approach is to recover the relational structure of web data. Here information about a particular book has been extracted and put into a relational table.

our focus is not on the spidering task, so we limit the collection of pages in a web site to those that can be spidered without navigating through web forms, although the extraction approach would be as applicable to pages that can reached via web forms as it is to pages that can be reached directly.

Typically, a web site is hosted on a particular domain, such as cs.cmu.edu or amazon.com, although some domains host multiple web sites and some other web sites are hosted on multiple domains. For example, it is common for internet providers to host separate web sites for their users and for large web sites to spread their pages across multiple domains, e.g., domains in multiple countries. In this work, a web site is a collection of pages that provide a consistent presentation of some data collection regardless of the domain or domains it is hosted on.

Our goal is to take a web site and turn it into a set of *relational tables*. A relational table is a set of *tuples* (or *records*). Each record in a table has a set of attributes and records in the same table have the same set of attributes. They can be related to other records through the use of references to the unique identifiers of other records.

Figure 1.1 shows how the tokens on a web page would be placed in a relational table. The partial view of the table displays three records, each containing the data from a single page; and four columns, each corresponding to some data field on the page.

One way to think about the site extraction problem is to consider the inverse problem of site generation: given relational data, generate a set of pages and links to present it as a web site.

1.2.2 Inverse Problem - Site Generation

As in many other structure discovery problems, site extraction has an inverse problem, namely site generation. The inverse problem has the following form in general: given structured data, present the data in a form appropriate for consumption (by humans, typically). Some examples of inverse structure discovery problems are voice synthesis from text and scene-generation from 3d-models. In this section, we will briefly examine site generation.

Site generation is the task of creating a web site to present data. As a concrete example, take the case of presenting weather information. We'd like to present current weather data, such as temperature, humidity, etc., for U.S. cities. A diagrammatic view of a such a site and the relation between its pages and the underlying data is shown in Figure 1.2. The weather data is best represented as a relational table "CityWeather" where the attributes are CityName, Temperature, Humidity and Wind. Generating an HTML page for a given city involves looking up the given city in CityWeather and decorating the result with appropriate HTML tags. We call the set of pages that are generated from individual rows of a relation table a *page type*.

Geographical data inherently has more structure than a flat list. Places are normally organized in a hierarchical fashion, from larger regions at the top of the hierarchy to smaller ones at the bottom, with the "world" as the root of the hierarchy. As is typically done, this hierarchy can be used to organize the weather site. For example, U.S. cities can be grouped under the state in which they are in. To represent this relation, we expand the relational model as follows. The states are stored in a new table "State" with attributes StateID, StateName, and Abbrv. The city table has a new attribute StateId to link each city to its state. For example, the StateId of Pennsylvania would be stored in the rows for Pittsburgh, Philadelphia, and Harrisburg among others.

Generating HTML pages for States involves retrieving data from more than one table. The first step is to look up the given state in States and find StateID, StateName and StateAbbr. Next, the list of cities that are related to StateID are found in CityWeather. Finally, the HTML page is generated for the given state using the values found in the previous steps. The state page shows the attributes of the state and the list of cities in that state. Each city listed on a state page can also be hyper-linked to the corresponding city page. Figure 1.2 depicts the structure of the weather site and its relation to the tables in which the underlying data is stored.

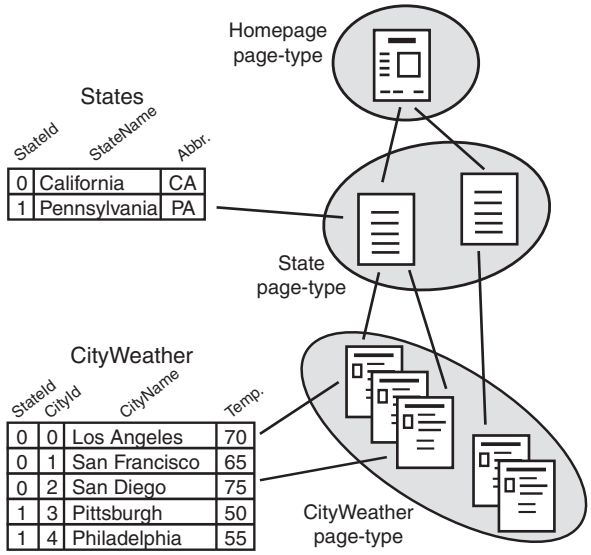


Figure 1.2: Relational Model of a Web Site — Two relational tables store the data presented on this web site. Each row of the States table corresponds to a page that displays information about a particular state. Similarly, each row of the CityWeather table corresponds to a page that displays information about a particular city.

Site extraction is the inverse of site generation. Given a set of HTML pages from a web site, the problem is to discover the relational form of the data. Unfortunately, the mapping from relational data to web pages cannot be mathematically inverted in the general case because it is clearly not a one-to-one mapping.

Clearly, not all elements of the web site are in relational form. For example, parts of the HTML page are generated by a program and are not stored in relational form and there is no way to know definitively which are which ahead of time. Also, more than one model can generate the same set web site, i.e., there is not necessarily a unique solution.

1.3 Thesis Statement

The problem of data extraction from structured web sites can be solved by developing experts that encapsulate human knowledge about various types of web structure and then combining these experts while maintaining a globally consistent model of the underlying data.

The approach is applicable to web sites where some combination of surface structures represent the underlying relational structure of the data and recognizers for these surface

structures can be encoded as experts.

1.4 Roadmap

In the following chapters, we first describe the principles behind our approach. Next, we develop a basic framework for combining multiple experts, a method for evaluating it and report our results. Then, we describe a more sophisticated framework which allows probabilistic hypotheses. We report on our experiments with the second framework, comparing it to the first framework in the web domain. Next, we show the generality of our approach by applying it in a second domain which is different from the web domain and compare our results to another research project. Before concluding, we describe related work in several different areas of research.

Chapter 2

Approach - Principles

In this chapter, we discuss our approach of combining heterogeneous experts at a high-level. In the following chapters, we describe two concrete frameworks that follow the principles of our approach.

2.1 Motivation

To demonstrate the need for multiple experts in attacking the data extraction problem, we first try to solve it using grammar induction and demonstrate the shortcomings of relying on one particular type of structure. We pick grammar induction because it has been used for extraction from the web by other researchers (e.g., [24], [3], Page Templates in [50]) and also because it exemplifies the issues that arise from focusing on a single type of structure.

The idea behind data extraction via grammar induction is that if an appropriate grammar that generates a set of web pages can be found, then the symbols of the grammar can be used to label the data fields.

As an example, suppose we are extracting data from the following three pages:

```
p1: <h2>Los Angeles</h2><p>Temperature: 70
p2: <h2>Pittsburgh</h2><p>Temperature: 68
p3: <h1>California</h1><p>Select a City:
```

A grammar induction algorithm that incrementally builds a grammar from examples might first correctly induce the following grammar after observing the examples p1 and p2.

```
P -> <h2>S1</h2><p>Temperature: S2
```

```
S1 -> Los Angeles | Pittsburgh
S2 -> 70 | 68
```

This grammar already achieves the extraction goal. The data fields are identified by non-terminals S1 and S2 and the corresponding data values as "Los Angeles", "Pittsburgh", "70", and "68". When the third page is considered, the "correct" grammar becomes less obvious. One possible grammar is the following:

```
P -> T1 | T2
T1 -> <h2>S1</h2><p>Temperature: S2
S1 -> Los Angeles | Pittsburgh
S2 -> 70 | 68
T2 -> <h1>California</h1><p>Select a City:
```

Another one is this:

```
P -> S1 <p> s2
S1 -> <h2>Los Angeles</h2> | <h2>Pittsburgh</h2> | <h1>California</h1>
S2 -> Temperature: 70 | Temperature: 68 | Select a City:
```

Clearly, making the right induction decisions requires extensive use of heuristics, ranging from utilizing syntactic clues (e.g., not allowing HTML tags as data values or distinguishing HTML elements h1 and h2) to understanding semantic relations (e.g., rejecting California in a rule that expands to Los Angeles and Pittsburgh or rejecting "Select a City: " in a rule that contains name value pairs.

Some of these heuristics can be implemented by pre-processing the input. For example, replacing names of well-known entities with their semantic classes as a pre-processing step will in general help the grammar induction algorithm. Unfortunately, some heuristics are hard to represent in this way. For example, the visual layout of a page contains useful clues about the underlying relational structure of the page in that relational tables are usually represented as lists on the page. However, it is not clear how these clues could be utilized in a grammar induction approach.

The small number of types of structure that grammar induction can use limits the types of grammars that can be practically induced. Thus, grammar induction approaches (e.g., [24], Page Templates in [50]) typically assume that the pages have been clustered according to page-type and all input comes from a single page type. This allows the grammar induction algorithm avoid having to generate "clustering" rules, such as `P -> T1 | T2` above. Unfortunately, correctly clustering the pages is a significant part of the site extraction problem.

A better approach to the problem of site attraction is to build a general framework that allows any number of structures to be analyzed and combined in the process of discovering the underlying structure of the data.

2.2 Structure Discovery

In this thesis, we use the term *structure* to mean any type of regularity or pattern that data exhibits. There are many different languages for describing structure, some extensively studied and formalized, such as formal languages, graphs, and relations, and some ad-hoc and specific to particular problems, such as URL patterns and page layout. Our goal is to be able to combine information that is present in different types of structure to find the hidden, underlying structure of data.

In *structure discovery* problems, the goal is to discover the hidden structure of data from partial and potentially noisy observations. Structure discovery problems arise because as structured data is transformed into observable forms, its regularity is also transformed. *Surface structure* is what remains of the underlying structure in the transformed data. Often, the surface structure appears in multiple forms because there are multiple transformations which are applied to the data and different transformations take the original regularity of data into different types of patterns.

Many AI problems can be cast as structure discovery problems. For example, computer vision aims to determine the hidden 3D structure of a scene from 2D observations (e.g., [42]). Some of the types of surface structure in this domain are edges, shadows, parallax effects, texture, etc. In record deduplication[79], the goal is determine sets of records that represent the same entity from the noisy attributes of the records. The hidden structure is the grouping of records and some of the common types of surface structure are acronyms and common misspellings. In protein structure prediction (e.g., [37, 52]), the goal is to find various types of structure, such as folds, from sequence data or direct physical measurements of the protein.

In structure discovery, data is assumed to have a particular type of structure. The discovery problem is then the search for a specific instance of structure within the given type of structure. For example, in our grammar induction approach above, we assume that pages are generated from a regular grammar, which is a particular type of structure. The structure discovery problem is to find the particular regular grammar that actually

generates the pages.

An important choice in solving structure discovery problems is the type of structure. Some types of structures are more general than others in that any instance of the specific structure is also an instance of the more general structure. For example, context-free grammars are more general than regular grammars and so any particular regular grammar is also a context-free grammar. The more general the type of structure, the more difficult structure discovery is. On the other hand, the more specific languages for describing structures may not be expressive enough to capture the underlying structure of the data.

In this work, we are mainly concerned with one particular type of structure discovery problem, namely the site extraction problem that we have introduced previously. In contrast to some of the harder AI problems, such as natural language processing where the underlying structure of data is not only hidden but its type is an approximation of some as-yet-unknown mental structures, the underlying data in site extraction is in fact *structured* in that most web sites store their data in relational databases, which are of course highly structured.

2.3 Tackling Structure Discovery Problems

One common way to attack structure discovery problems is to first pick one type of structure as the underlying structure of data and try to find an instance of this structure for the given data. This is what we did earlier in the chapter when we used grammar induction to try to solve the site extraction problem.

For a given type of structure and a particular domain from which data is collected, it is usually possible to develop an algorithm, usually incorporating heuristics, to induce the particular structure of given data. For example, certain types of regular grammars can be induced by finding common substrings of samples.

The main problem with focusing on one particular type of structure is that it is difficult or impossible to pick the type of structure with the appropriate level of generality. Specific types of structures are preferable as the discovery problem is easier with more specific structures and for that structure are likely to be more accurate[78]. For example grammar induction with regular languages is easier than with context-free languages. However, the structure cannot be so specific that it is not able to represent all possible patterns in the underlying data. Unfortunately, as the type of structure is allowed be more general,

the structure discovery problem becomes more and more under-specified, as was the case in our grammar induction example, and some other heuristics of choosing an "optimal" structure is needed.

Alternatively, the specific type of structure can be extended to be more general, for example allowing certain types of context-free productions in a regular language. Unfortunately, such extensions require ad-hoc heuristics which require an ad-hoc patch to the heuristic induction algorithm and doing this becomes increasingly difficult as the number of extensions increases.

In this thesis, we propose a new approach to tackle structure discovery problems. In our approach, instead of picking a particular type of structure into which we fit the data, we look for many different types of specific structure in the data and then combine the information we have gained in the process in a more general structure.

An intuitive way to handle multiple types of specific structure is to develop an expert for each type of specific structure. Each expert is then responsible for finding particular types of structure in the observed data and relating this structure to the more general structure.

For example, in the computer vision domain, a "boundary" expert can detect *boundaries* which separate pixels that are formed by separate objects in the actual scene and relate this information into objects in 3D. Detecting boundaries requires interpreting the brightness, color and texture variations in an image (e.g., [54]). The complexity of understanding a particular type of clue, such as boundaries, is captured by an expert.

2.3.1 Multiple Heterogeneous Experts

Building experts which deal with arbitrary types of surface structure naturally leads to a heterogeneous collection of experts. This is because as we discussed earlier, each type of structure has an associated algorithm that induces the particular structure from the data. These algorithms typically are highly-specific to the types of structures that they are working on and thus heterogeneous. For example, in the site extraction domain, one expert might search for simple regular grammar patterns in data fields whereas another expert might search layout patterns on pages. These experts are heterogeneous in that not only do they use different characteristics of the data, but the patterns they find are in a sense in different languages.

Even though a set of heterogeneous experts elegantly handles multiple types of specific

structure, it introduces a new challenge: If the experts are all working independently and finding arbitrary types of patterns, then combining these patterns becomes non-trivial. For example, if the regular grammar expert above finds particular grammar rules and a set of values generated by these rules, whereas the layout expert finds DOM elements that are arranged in a column, it is not clear how these patterns can be combined so that more of the underlying structure is revealed.

To solve the problem of combining the outputs of heterogeneous experts, we introduce a common language in which each expert can express its patterns. Experts are still allowed to have heterogeneous views of the problem but output their hypotheses in a language that is globally known. In the following chapters, we experiment with two languages. The first language allows the experts to output their hypotheses in terms of primitive statements. The second language extends the first by allowing the experts to assign confidence scores to these statements.

2.3.2 Combining Heterogeneous Experts

As mentioned earlier, surface structure arises when the underlying structured data is transformed into observable forms. Multiple transformations take the underlying structure into multiple types of surface structure. This process clearly implies a dependence of surface structures on the underlying structure. The dependence between multiple surface structures and the underlying structure, when combined with the independence of transformations, lead to an interesting and useful property: Even though from a single surface structure, many different but equally valid underlying structures can be induced, the number of valid structures shrinks rapidly when multiple surface structures are taken into account. In other words, maintaining consistency across multiple surface structures allows the hypotheses to be pruned to a degree that is not possible with individual experts.

This is best demonstrated with an example: Suppose on a web site, a product page mentions the price of the product and the shipping cost. A "price" expert that interprets a currency symbol followed by a decimal number will interpret both the price of the product and the shipping cost as valid prices. Assuming each product page has exactly one value for price, this expert leads to two plausible interpretations for each page: one where the product price is correctly identified and one where the shipping cost is incorrectly identified as the price. Now, suppose we add a second expert that understands that the range of prices are much larger than the range of shipping costs. This expert will be able to

eliminate the second interpretation on pages where the shipping cost is relatively small to the price of the product. There is still a problem, though, because if the product price is within the same range as the shipping cost, the second expert cannot help. Next, suppose we add a third expert that understands that a single site normally lists costs in the same order (e.g., product price comes before shipping cost) on all product pages. The third expert will be able to eliminate the remaining incorrect interpretations by taking into account the interpretations of the second expert on pages where product price and shipping cost are more easily identified.

2.3.3 Experts vs. Complex Models

Experts encode and encapsulate domain knowledge about the relation between underlying structure of the data and its surface structure. An alternative approach to solving structure discovery problems is to use "simple" features and induce the relation between underlying and surface structures as part of the discovery process.

As a concrete example, let's consider taking advantage of page layout in discovering the relational structure of data on a web page, while using simple features. Our goal is to learn from training data the statement that data from a relational column is typically displayed so that data elements line up on a vertical line. To represent page layout, we use as features the x and y coordinates of data elements on the screen. The induction step then is responsible for learning the concept of alignment in terms of x and y coordinates, perhaps by searching through a space of equality constraints on coordinates. This is where model complexity arises: having simple features necessitates having a space of models where complex relations between simple features can be expressed.

Compare the approach above to the one using "experts". If it is known that alignment on the screen is a possible indicator of underlying relational structure, it can directly be represented as a "rich" feature (perhaps, as a boolean feature among pairs of data elements). This simplifies the induction step greatly because the model no longer needs to represent complex concepts that involve x and y coordinates.

In general, there is an inverse relation between rich features and complex models: Richer features require simpler models whereas complex models can utilize simpler features. The question then is whether it is better to solve a given structure discovery problem using "experts" and a simpler model or using simple features and a more complex model. In making this decision, various trade-offs need to be taken into account. In particular,

richer features by definition restrict the system to consider only the concepts that are already represented by features, whereas complex models tend to be more general in the concepts that they can induce. With this in mind, we can describe a key characteristic of structure discovery problems which are suited to expert based approaches: A structure discovery problem is a good candidate for an expert based approach if the set of transformations that map the underlying structure into the observed structure are well-known and each transformation can be defined independent of each other.

The web site extraction problem exhibits the characteristic above. First, the set of transformations that map the underlying structure to the surface structure are well-known (e.g., use of HTML templates to generate pages from relational tuples, use of links to connect pages for related tuples, laying out pages so that relational tables are displayed as tables, and so on). Second, individual transformations are independent (e.g., links are independent of layout and independent of tables). Contrast this to a domain, such as protein structure discovery, where a complex model might be more suitable. In such domains, the underlying structure and the observed structure are interlinked through as-yet unknown or only partially known relations. Thus, these relations are difficult or impossible to encode as experts and approaches based on complex model are more suitable than expert based approaches.

2.4 Summary

In this chapter, we described the main principles of this thesis. In particular, we discussed structure discovery problems in general and why these problems are best attacked with a set of heterogeneous experts. In the following chapters, we describe two frameworks we build on these principles.

Chapter 3

CHEX: Clustering with Heterogeneous Experts

3.1 Overview

In this chapter, we focus on developing a basic probabilistic framework in which hypotheses from heterogeneous experts are combined. We develop the framework for tackling a specific data extraction problem, which we call site extraction: Extract all data that is available on a given set of pages from a web site.

We start with a set of pages and the links between them, which we collect by spidering. For example, for an online bookstore, the set of pages would include *list pages* that present lists of books grouped under various categories and *detail pages* that contain information about individual books, such as titles, author names, ISBNs, reviews, etc.

We assume the sites we are extracting data from are generated from relational tables. This assumption is not limiting for two reasons: First, a large percentage of web sites are in fact generated from data that is stored in relational tables. This is because relational databases have become the norm for storing and querying collections of data since their introduction around 1970[15]. Second, web sites that are not generated from relational tables can sometimes be interpreted as if they are generated from relational tables.

Our goal is to find clusters of data such that each cluster corresponds to a column of one of the relational tables. On an online bookstore, we expect to find, among many others, a cluster of ISBNs, as ISBNs are commonly used to identify books.

The relational representation of data on a given web site is not necessarily unique or

even well-defined. Our goal is to find some relational representation such that further processing of data becomes much simpler. This is similar to the goal of clustering in general, which is to aid analysis of large amounts of data. Further processing might involve other automated processes (such as labeling columns) or human input (selecting relevant columns of data for the task at hand).

In finding the relational representation of a web site, there are three stages of processing. In the first stage, experts analyze the set of pages and links, and output a set of hypotheses about the structure of the underlying data. In the second stage, clusters of data that is marginally consistent with respect to the hypotheses is found. In the third stage, the clusters are converted to relational form.

In the following sections, we will define the expert framework, the hypotheses language, what it means for a clustering to be optimal with respect to a set of hypotheses, and our approach for searching optimal clusterings. We will also describe the experts we use in detail and our method for converting clusters to relations.

3.2 Expert Framework

In many machine learning algorithms, samples are represented by their "features"[27]. For example, in clustering points on a 2-dimensional space, each point sample has two features, its x-coordinate and y-coordinate. In a document classification problem, one way to represent documents is by turning each possible word into a binary feature such that the value of the feature is 1 when the document contains the word and 0 otherwise.

Features simplify the learning problem by reducing it to one that is on features rather than the potentially more complex samples. In particular, each feature is typically a function of the sample. Given a sample, the value of a particular feature can be computed in isolation from the rest of the samples.

For the structure-induction task, the reduction to feature-space is sometimes too much of a simplification and makes the learning task too hard. In particular, some of the relations between features are lost. One way of recovering some of these lost dependencies is by post-processing the features, such as normalizing feature values or selecting important features.

As in learning algorithms, clustering can be hindered when samples are mapped into feature-space. This is especially the case because most clustering algorithms rely on the

notion of distance between samples[27] and some distance metrics in the sample-space cannot be represented as a distance metric in the feature-space. Consider clustering strings such that words that have similar spellings are clustered together (e.g., in building a spell checking application). String-edit distance would be a useful measure for this problem, but it can only be computed on pairs of strings directly and not on simpler features of the string.

Within the clustering domain, one way to improve the expressivity of features is to define features on pairs of samples rather than individual samples. This leads to pairwise-clustering (e.g., [40]) which allows metrics such as the string-edit distance to be computed as a feature as in [25].

The generalization of features from the domain of individual samples to pairs of samples can be taken further and leads to the concept of experts. Whereas features are usually simple functions that map one or two samples into feature values, experts are arbitrarily complex processes, which may refer to domain knowledge, or analyze the problem globally, before outputting their decisions.

Note that in defining the output of experts, we restricted the hypotheses to those that can be computed given a single element of the solution space. Clearly, the range of all possible hypotheses is much larger. As a simple example, consider an expert that generates the hypothesis that a particular problem instance has a unique solution. Such a hypothesis cannot be verified for individual elements of the solution space.

The elements of the solution space have *structure* in that hypotheses can be evaluated within the context of each. Contrast this to a non-structured space where the elements are atomic. In a non-structured space, the only possible type of hypothesis is whether the element is a solution or not. The space of clusterings is an example of a structured space in that many hypothesis can be stated and tested within a given clustering, e.g., "element e_1 and e_2 are in the same cluster", "number of clusters is at least 3".

An important observation is that the set of hypotheses output by an expert define a subset of the solution space where within each element of this subset all the hypotheses are true. If this subset contains more than one solution, then the expert has not been able to determine a unique solution. If the subset is empty, then the hypotheses output by the expert is inconsistent in that there are no solutions in which all the hypotheses can be satisfied.

The above observation leads to a straightforward, albeit fragile, framework for combining experts: The set of solutions is the intersection of all the subsets defined by the

hypothesis sets. Alternatively, the set of solutions is the subset where the hypotheses from all the experts are true.

The problem with this approach is that in practice the hypotheses from multiple experts is likely to be inconsistent even if each expert generates self-consistent sets of hypotheses. In particular, any hypothesis that is incorrect (i.e., false in solutions) will make the set of hypotheses inconsistent if the remaining hypotheses are all true only in the solution set. Thus, this type of approach is highly sensitive to any incorrect hypotheses generated by the experts and will fail to find solutions unless the experts generate only correct hypotheses.

An interesting case where an expert generates inconsistent hypotheses is when the expert works on parts of a solution without considering the global consistency of the solution. For example, in the crossword domain, an expert that works on individual clues can easily generate hypotheses that when applied within the constraints of the grid are inconsistent. If one expert requires 5-across to be either "LET" or "NET" and another expert requires 5-down to be "RAIL" or "SAIL", clearly there are solutions in which both sets of hypotheses can be satisfied.

As an alternative and more robust approach, we define a probabilistic model of the process of hypothesis generation so that for a given element in the solution space, each hypothesis is assigned a probability. This allows us to search for solutions such that the probability of the set of hypotheses generated by the experts is maximized. Note that unlike the first framework we described where the set of hypotheses partitions the space into solutions and non-solutions, the probabilistic approach orders its elements such that all are candidate solutions, but some are more likely to have generated the observed set of hypotheses. This is true even when the hypotheses may be inconsistent. Before we describe the probabilistic model in detail, let us first define the particular hypothesis language we will use for clustering.

3.3 Hypothesis Language

We have already restricted the scope of hypotheses by requiring each to be testable given any element of the solution space. Additionally, we desire the language to have the following properties:

- It should be expressive enough for arbitrary experts to output their hypotheses.
- It should be simple enough so that multiple sets of hypotheses can be combined

effectively.

In the clustering domain, this is achievable even by allowing only one type of hypothesis: one which states that a particular pair of samples is in the same cluster. We call such a hypothesis a *hint*.

Using this language, experts that evaluate pairs of samples can directly output their hypotheses as hints. Such experts typically evaluate the similarity of two samples and if that similarity is above some threshold, hypothesize that the pair is in the same cluster.

More complex hypotheses are expressible, too. For example, an expert that finds clusters of samples outputs a hint for each pair of samples that is in the same cluster.

3.4 Probabilistic Model

Our ultimate goal is to find the most likely clustering given the set of hypotheses generated by the experts. We do this by defining a probabilistic process of hint generation.

Probabilistic approaches to document clustering are not new. The goal in document clustering in general is to find clusters of documents that are within the same topic. In the typical approach, each document is represented by a vector of word counts, normalized by one of a variety of methods, such as TF-IDF[64]. The probabilistic approaches model the distribution of these vectors. A common approach to do this is to model the distribution of words as a mixture model. In a mixture model, the word distribution is modeled as a mixture of distributions that are conditioned on particular topics. The model can be used for classification when topics and the parameters of the model are known or computed beforehand[56]. The model can also be used for clustering by applying statistical inference techniques such as EM to find locally optimal assignments to the unknown parameters from observed data[6, 39].

The main focus of document clustering research is to find probabilistic models that better fit data and to develop methods to estimate unknown parameters of these models accurately and efficiently. For example, [1] improves on the bag-of-word representation by using distributions that better model the dependence between multiple occurrences of a word within a document. In contrast to the focus of this type of research, our focus is making use of knowledge encoded within sophisticated experts and not on the probabilistic model. Thus, we use a simple probabilistic model which serves to combine the output of experts in a principled framework. We empirically assign values to the parameters of the

model and leave as future work inferring these from observed data.

To find the most likely clustering given the set of hypotheses generated by the experts, we need to compute $P(C|H)$ where where H is the set of hypotheses and C is the clustering. We follow the usual approach of using the Bayes rule to expand this probability as follows:

$$P(C|H) = \frac{P(H|C)P(C)}{P(H)}$$

. We assume that the prior distribution $P(C)$ is a discrete uniform distribution, i.e., $P(C = c_i) = k$ for all clusterings c_i and for some constant k. Combining this assumption with the observation that $P(H)$ is a constant for a given set of hypotheses, finding the clustering that maximizes $P(C|H)$ is equivalent to finding the clustering that maximizes $P(H|C)$.

Like in naive Bayes classification, we then make an independence assumption, namely we assume that each hypothesis is independent of other hypotheses given the clustering. This leads to the following expansion:

$$P(H|C) = P(h_1|C)P(h_2|C)P(h_3|C)...P(h_n|C)$$

where h_i is in H.

The next step is to define $P(h|C)$ where h is a single hypothesis, in other words a hint. We do this by defining a stochastic process of hint generation. This process has two steps:

1. Pick a cluster at random from C. The probability of picking a particular cluster is $1/N_C$ where N_C is the number of clusters. Call the selected cluster c_0.
2. Pick a pair of elements from c_0 at random. The probability of picking a particular pair is $1/N_P$ where N_P is the number of distinct pairs in c_0. Call the selected pair h.

Thus, the probability of generating h from C, $P(h|C)$, is $1/N_C \times 1/N_P$ where N_C is the total number of clusters in C and N_P is the number of pairs in the cluster which contains the pair of samples of the hypothesis h.

The following table shows the probability of generating the hypothesis that the elements A and B are in the same cluster for all the possible clusterings of three elements A, B, and C. Given only one hypothesis that A and B are in the same cluster, the most likely clustering based on the above process is AB,C. When the number of samples to be clustered is large, the number of clusterings is too large to be enumerated. Thus, an approximate search approach is necessary. In the next section, we describe how we search for the most likely clustering using the leader-follower algorithm[27].

Clustering	$P(h_{A,B})\|C)$
ABC	$1/1 \times 1/3 = 1/3$
AB,C	$1/2 \times 1/1 = 1/2$
AC,B	0
BC,A	0
A,B,C	0

The process assigns a probability of zero to hints that are inconsistent with respect to a given clustering. A hint is inconsistent with a given clustering if the pair of samples is not in the same cluster. The process described above has no way to generate such inconsistent hints. This implies that when any hint in the set of hypotheses is inconsistent with a clustering, the overall probability of the clustering is 0. Thus, when the set of hypotheses is inconsistent, i.e., there is no clustering such that all hypotheses can be satisfied, then this probabilistic model cannot pick a clustering that is "most consistent" with the hypotheses as it assigns a zero probability to all clusterings.

We extend the model by incorporating a simple model of noise. To do this, we add a new way of generating hints that is independent of the given clustering. The new way of generating hints simply picks a pair of samples uniformly from all possible pairs, and so allows all pairs to be generated as "noise" with some probability. We then extend the overall process so that it either uses the normal steps of generating hints or the steps to model noisy generation. The latter set of steps is followed with some small probability. The new process has the following steps:

1. Pick whether the hint will be generated using the noise model or the basic model as above.
2. If the hint will be generated using the noise model, pick a pair uniformly among all sample pairs. The probability of picking a particular pair is $1/N_P$ where N_P is the number of all sample pairs.
3. Otherwise:
   (a) Pick a cluster at random. The probability of picking a particular cluster is $1/N_C$ where N_C is the number of clusters. Call the selected cluster c_0.
   (b) Pick a pair of elements from c_0 at random. The probability of picking a particular pair is $1/N_{c0}$ where N_{c0} is the number of distinct pairs in c_0. Call the selected pair h.

In the new process, the probability of a hint given a clustering comes from two factors: a small contribution from noisy generation and a large contribution from normal generation. For hints that are inconsistent with the current clustering, the factor from normal generation is zero, but there is still a non-zero contribution from noisy generation. This allows the model to assign non-zero probabilities to all clusterings even when some hypotheses are inconsistent. The inconsistent hypotheses simply reduce the probability of a clustering.

3.5 Clustering

After the hypotheses are gathered from the experts, the next step is to find the most likely clustering. The number of clusterings is at least exponentially large in the number of samples[1], so a linear search through all clusterings is not possible. Instead, we use the leader-follower algorithm, a standard clustering algorithm[27] which runs in linear time in the number of samples.

The leader-follower algorithm considers each sample in turn. The first sample is placed in a cluster of its own. Then, for each sample, the nearest existing cluster is located. If the distance to this nearest cluster is below some threshold, the sample is added to that cluster. Otherwise, the sample starts a new cluster.

Compared to some other methods such as the agglomerative clustering algorithms which run in quadratic time, the leader-follower algorithm is fast but makes decisions given very limited knowledge especially at early stages where it has seen only a few of the samples. In the approach described in this chapter, we utilize the leader-follower algorithm, but other approaches are certainly possible. In later chapters, we experiment with more sophisticated clustering algorithms.

The "distance" between a sample and a cluster is the change in probability of the set of hints between the current clustering and the clustering when the sample is added to the given cluster. As the leader-follower algorithm builds the clusters, the clustering is only partial, i.e., the unseen samples have not been assigned to any clusters. The probabilistic model cannot assign probabilities to hints that refer to yet unseen samples, so during the

[1]The number of clusterings for a sample set of size n is given by *Bell numbers*[63] which satisfy the recurrence $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$[53]. From the recurrence, it follows that $B_{n+1} > \binom{n}{n-1} B_{n-1} = nB_{n-1}$. Thus, increasing the size of a sample set of size $n-1$ by 2 samples increases the number of partitions by at least a factor of n. This means the number of partitions is at least exponential in the size of the sample set.

clustering process, we only calculate properties for hints that refer to the currently clustered samples. To determine the change in probability, we consider the clustering where the new sample is in a new cluster of its own and evaluate the change when the sample is added to one of the existing clusters.

$$distance(s, c) = P(H|C_s) - P(H|C_m)$$

where s is a sample, C_m is the current clustering but with s added as a new singleton cluster, c is a cluster in current clustering, and C_m is the new clustering after s has been merged into c. Note that the change in probability is not a "distance" in the formal sense. Whereas distance metrics are non-negative and satisfy the triangle inequality, the change in probability can be negative and does not satisfy the triangle inequality. However, it is still able to serve as a distance measure within the context of the leader-follower algorithm, which only depends on the distance score to make local decisions. In particular, the leader-follower algorithm uses this score to choose which cluster to assign a given sample by comparing the scores for different clusters while holding the sample constant. This process makes no assumptions about the distance measure.

Adding a sample to an existing cluster affects the overall probability in two ways: First, any hints between the current sample and the samples in the cluster are assigned higher probabilities as the generation process for those hints switches from noisy (step 2) to normal (step 3). Second, the hints between samples in the cluster are assigned smaller probabilities as the number of potential sample pairs increases (step 3b).

Other standard algorithms can also be applied to the clustering problem we have defined. Here we chose the leader-follower algorithm for its simplicity as we focus on combining heterogeneous experts that share a common language. Once the clusters are established, the leader-follower algorithm is optimal in the sense that its decision is based only on the similarity of the existing clusters to the next sample. However, this also makes it sensitive to the ordering of the samples. As an example, suppose the algorithm first sees sample s_0 which becomes the first cluster. When it processes the next sample s_1, the decision to place s_1 into the first cluster is based only on the similarity of s_0 and s_1. If s_0 and s_1 are not similar, then s_1 is put in a new cluster. Suppose now s_0 and s_1 are processed after some clusters have been established and s_0 is put in cluster c_0. It may well be the case that s_1 is still placed in c_0. This is because even though s_0 and s_1 are not similar, they may be considered similar when their neighbors in c_0 are taken into account. To address this issue, in the following chapters, we look at a second technique, hierarchical agglomerative

clustering[27].

3.6 Finding Data Clusters

Our main goal is to cluster tokens such that each cluster corresponds to one column of a relational table. In doing this, it is also convenient to cluster the pages themselves, in order to identify page types. Thus, we have both page clusters and token clusters with the following relation between page clusters and token clusters: Each page cluster is associated with one or more token clusters, such that the tokens of any page within the page cluster are all contained in the token clusters associated with that page cluster. More formally, we let $child(c_p)$ be the set of token clusters associated with page cluster c_p such that if t is a token on page p and $p \in c_p$, then there is a token cluster $c_t \in child(c_p)$.

For example, suppose we are working on the weather site which contains a home page listing all the states, state pages listing all the cities in that state, and weather condition pages that display the current conditions in a particular city. When we find the page clusters for this site, if the clustering matches our natural grouping, we would expect to end up with three clusters: one for weather condition pages, one for the state pages, and one containing just the home page. The token clusters for the weather condition page cluster might include a cluster for city names, another for low temperatures, and another for high temperatures. In Figure 3.1, two page clusters and the four token clusters that they are associated with are shown.

In the previous sections, we described the probabilistic framework in which we evaluate the clusterings. In those sections, we assumed there was one set of samples and one set of hints. Since now we are clustering both pages and tokens we need to extend the framework. To do this we consider the sub-problems independently. Thus, we have a page clustering problem where we would like to find the most likely page clustering given a set of page hints and we have a token clustering problem where we would like to find the most likely token clustering given a set of token hints. Following the approach described previously, a *page hint* is a hint indicating that two pages are in the same cluster and a *token hint* is a hint indicating that the two tokens are in the same cluster.

We apply the leader-follower algorithm at both the page clustering level and the token clustering level. To do this we process the input one page at a time. To determine the page cluster in which the current page will be placed, we try placing the page in all possible page

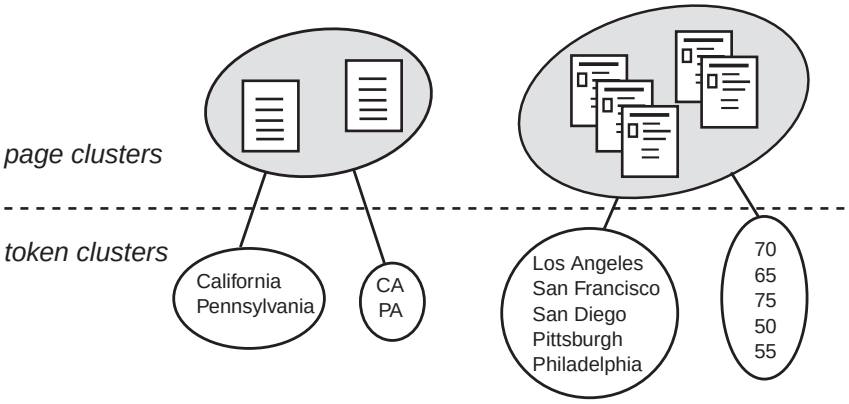


Figure 3.1: Clusters of Pages and Tokens — Pages and tokens are clustered at the same time while maintaining the constraint that tokens from two pages can be in the same token cluster only if the pages are in the same page cluster.

clusters. When we place the page in a page cluster, we apply the leader-follower algorithm to all the tokens of that page, clustering the tokens into the page clusters associated with the page cluster. Clearly, each trial of placing the current page in a page clustering results in a new clustering. We evaluate the set of hints for each such clustering. The clustering that improves the overall probability of hints is kept as the starting clustering for the next iteration. Algorithms 1 and 2 contain an implementation of the leader-follower algorithm for clustering pages. Note that in finding the cluster to which a page is going to be added, a separate instance of the leader-follower algorithm is run on the tokens of the page.

Inputs: pages, hints, threshold
Outputs: clustering
  page_clustering = $\emptyset$
  for all page in pages **do**
    prob_change = 0
    if page_clustering is not empty **then**
      best_clustering = best_page_clustering(page, hints, page_clustering)
      prob_change = prob_change(hints, page, best_clustering, page_clustering)
    end if
    if page_clustering is empty or prob_change $<$ threshold **then**
      create new page cluster c_new with page in c_new
      add c_new to page_clustering

```
    else
      page_clustering = best_clustering
    end if
  end for
  return page_clustering
Inputs: page, hints, clustering
Outputs: clustering
  best_clustering = ∅
  for all c in clustering do
    c′ = leader_follower_tokens(page, c, hints)
    clustering′ = clustering - c + c′
    if prob(hints | clustering′) > prob(hints | best_clustering) then
      best_clustering = clustering′
    end if
  end for
  return best_clustering
```

3.7 Web Experts

One obstacle in clustering data items on a given web site is determining which sequences of characters constitute atomic data items. In this work, we avoid this problem by having each expert determine boundaries according to their particular needs.

This is a problem similar to word or sentence boundary detection in natural language processing in that boundaries can be ambiguous until after discovering the higher structure of the data, e.g., parsing the text or finding the relational form of the web data, but discovering the higher structure can only be done after the boundaries are determined. For example, an expert working on the parse tree of an HTML document may use HTML tags to determine boundaries in the text. For example, from "ISBN: 11112222" an expert may pick up "ISBN: 11112222" as a single token and not divide the text into smaller tokens. After all experts are done with their processing, the union of all token boundaries that experts use in their output determines the "global" token boundaries, thus the set of data items. For example, if the output also contains "11112222" in addition to "ISBN: 11112222", then the data items are "ISBN: " and "11112222" because there are three token

boundaries: right before "I", between the space and "1", and right after "2". Additionally, output that originally referred to the single token "ISBN: 11112222" is updated to refer to the token sequence "ISBN: ", "11112222". In addition, hints whose tokens are split by this process, are rewritten as multiple hints over all new pairs. In other words, a hint on (s, t) is rewritten as multiple hints $(s_1, t_1), (s_1, t_2), ..., (s_1, t_n), (s_2), (t_2), ..., (s_m, t_1), ..., (s_m, t_n)$ if s is split into $s_1, ..., s_m$ and t into $t_1, ..., t_n$.

In applying our framework to the site extraction problem, we combine hypotheses from four experts, which look for the most common types of web site structure. One of the four experts generates page level hints and three generate token level hints. First we describe the page level expert that analyzes URLs.

- **URL Patterns**: On many web sites, URLs clearly identify page types. For example, on bookpool.com, pages listing the results of a query contain the query string "ss?qs=" and pages displaying the details of a page contain the string "sm/" where the text in brackets is a replaced with actual values. Here are some actual URLs from bookpool.com.

  http://www.bookpool.com/ss?qs=clustering&x=0&y=0

  http://www.bookpool.com/ss?qs=learning&x=0&y=0

  http://www.bookpool.com/sm/1590598296

  http://www.bookpool.com/sm/0131882228

  To utilize the URL structure, we'd like to find pairs of URLs which are similar and then generate a page hint to indicate that the pages for these URLs are likely to be in the same cluster. To determine whether two URLs are similar, we check if the URLs have been generated from the same *template*. A template is a string where data fields have been replaced with placeholders. By substituting actual data values into the placeholders, a template can be used to generate a set of strings. In particular, we define a template as a sequence of alternating *stripes* and *slots* where stripes are string constants and slots are placeholders for data.

  The definition above allows any two URLs to be generated from a template even if the URLs are not similar, simply by having a template with one "large" slot that takes in the whole URL as data. To make use of URL structure, we need to be able to distinguish trivial templates and templates that indicate similarity. Intuitively, a good template is one where the stripes of the template cover a high percentage of

the characters of the URLs. For example, the template for the first two URLs above, which is shown below, has a coverage of 91%. The slots of the template are shown as boxes.

http://www.bookpool.com/ss?qs=[0]e[1]r[2]ing&x=0&y=0

In contrast, the template for the first and the third URLs has a coverage of 59%:

http://www.bookpool.com/s[0]

Thus, our URL pattern expert works as follows: Consider each pair of pages. For each pair, build a template for their URLs and compute the coverage of the stripes of the URL. If the coverage is above some threshold, then generate a page hint for that pair of pages.

Note that the URL comparison done by this expert is simplistic in that it does not take into account the inherent structure that URLs have. For example, a difference in the host name (e.g., www.bookpool.com) is more significant than a difference in some identifier (e.g., 013882228). This shortcoming can be addressed by using more sophisticated measures of string-similarity[17] that can take into account the distribution of samples or can be trained with labeled examples[16]. In related practical work, we have also experimented with improving the URL expert so that various common components that appear in URLs, such as host names, unique identifiers, dates and so on, are identified and compared appropriately. In this dissertation, we report results obtained with the basic URL expert.

The following list describes the token level experts in more detail:

- **Templates**: Perhaps the most prevalent type of structure on web pages arises from templates. We have already defined templates in describing the URL pattern expert. Here, we use templates as generators for groups of web pages where data fields have been replaced with placeholders. To generate a set of web pages corresponding to a set of records, the template is instantiated for each record. Instantiating a template involves replacing the placeholders with appropriate values from the record.

  It is not surprising that templates are a common type of structure, because they allow producers of information to provide a familiar context to consumers of information with minimal effort. The unchanging parts of the template provide context for the data fields. The following example illustrates the use of templates in detail.

  The following two HTML segments contain weather information for two cities. Af-

ter spending a few seconds to learn the structure of the following HTML segment, it is easier to find the same weather information for a different city in the second segment.

```
<TD VALIGN=MIDDLE ALIGN=CENTER CLASS=obsInfo2 WIDTH=50%>
  <B CLASS=obsTempTextA>54&deg;F</B>
</TD></TR>
<TR><TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>Partly Cloudy</B>
</TD>
<TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>Feels Like<BR>54&deg;F</B>
</TD></TR>


<TD VALIGN=MIDDLE ALIGN=CENTER CLASS=obsInfo2 WIDTH=50%>
  <B CLASS=obsTempTextA>77&deg;F</B>
</TD></TR>
<TR><TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>Fair</B>
</TD>
<TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>Feels Like<BR>75&deg;F</B>
</TD></TR>
```

Like the URL pattern expert, the template expert is concerned with a specific kind of template where a template is a sequence of alternating slots and stripes. A template for the two segments above is shown below. The slots of the template are the labeled boxes.

```
<TD VALIGN=MIDDLE ALIGN=CENTER CLASS=obsInfo2 WIDTH=50%>
  <B CLASS=obsTempTextA>[actualTemp]&deg;F</B>
</TD></TR>
<TR><TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>[currentCondition]</B>
</TD>
<TD VALIGN=TOP ALIGN=CENTER CLASS=obsInfo2>
  <B CLASS=obsTextA>Feels Like<BR>[feelsLikeTemp]&deg;F</B>
</TD></TR>
```

The idea behind the template expert is to first find a template for a pair of pages and then use the slots to determine which data items are likely to be in the same cluster. The first step in doing this is a to determine a template for a set of pages, a simple form of grammar induction. We describe the details of the induction algorithm in a later chapter. Once the template is determined, the data values that have been used to generate specific pages from this template can be easily determined. Then, for each pair of values that fall into the same slot, the template expert generates a token hint. For example, in the example above, the slot *actualTemp* has been assigned the value 54 to generate the first segment and 77 to generate the second segment. Thus, the template expert generates a token hint for the pair $(55, 74)$, as well as for the pairs $(Fair, PartlyCloudy)$, $(54, 75)$.

- **List Structure**: Another common type of structure of web pages is tabular structure. Tables are used when the results of a query that retrieves more than one record are to be displayed. Multiple records are displayed in a list whose rows are generated by iterating over the records.

  Just as a page shares a common structure with pages of the same page-type, each row of a list shares a common structure with the other rows in the same list, and the template idea applies equally well to the rows of a list as it does to pages. Let us call the template representing the common structure of rows in a list a *row template*. We extend the basic template model so that slots are either *content slots* as before, or *list slots*, which are containers for row templates.

  As an example, let's create a template to generate HTML segments like the following segment:

```
<h2>Cities in Pennsylvania</h2>
<ul>
  <li>Aaronsburg</li>
  <li>Abbottstown</li>
  <li>Abington</li>
  <li>Ackermanville</li>
</ul>
```

  The outer template generates the first, second and last lines of the segment and "calls" the inner template (represented by a double box) to generate the elements of the list:

```
<h2>Cities in [state]</h2>
<ul>
  [cities]
</ul>
```

The inner template, `cities`, simply generates one row of the list.[2]

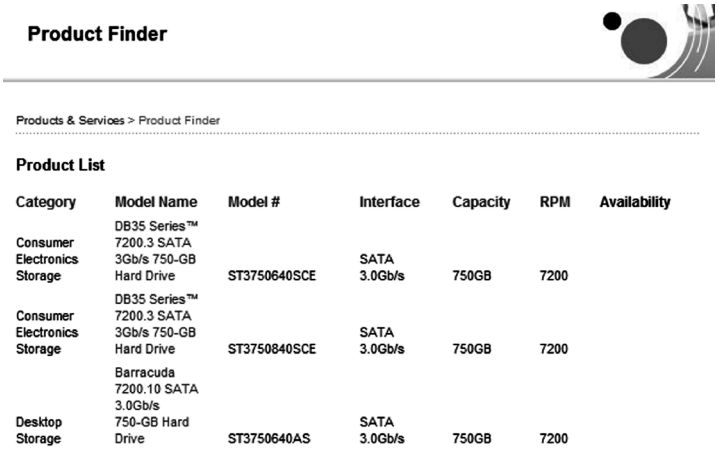
```
<li>[city]</li>
```

Like the template expert, the list structure expert first finds templates, then finds the values that have been substituted into the slots, and finally generates token hints for tokens that fall into the same slot. In the example above, the `city` slot is instantiated with values from the set {Aaronsburg, Abbottstown, Abington, Ackermanville}. Thus, the list structure expert generates token hints for all the pairs: $(Aaronsburg, Abbottstown), (Aaronsburg, Abington), ...$

- **Layout**: Another common type of structure on web pages is that values of the same field are placed at the same horizontal position. This structure is more common within a single page, where values of the same field are aligned in the same screen column much like the columns of a spreadsheet. The structure is also appears across multiple pages for fields when to the left of the field only fixed-width items such as images and labels are present. Figure 3.2 shows an example where alignment of values in the list matches perfectly with the underlying structure of the data.

  The layout expert uses a web browser (Internet Explorer, IE for short) to find positional information. To render an HTML document on the screen, IE parses the HTML document into a DOM tree, and lays out the nodes of this tree according to the HTML specifications. In the process, IE assigns x and y-coordinates to nodes that contain text elements. The layout expert queries IE to retrieve position informa-

[2]In practice, more information needs to be specified before pages can be generated. In particular, in addition to the slots being linked to the underlying data source, a method for determining the set of cities for a particular instantiation of `state` needs to be specified. To make this concrete, suppose the data source is the one shown in Figure 1.2. The slot `state` is linked to the "StateName" column of the "States" table. The slot `city` is similarly linked to the "CityName" column of the "CityWeather" table. The outer template is instantiated for every row of "States" whereas the inner template is instantiated for rows of "CityWeather" where the "StateId" column of "CityWeather" matches the "StateId" column of "States" for the current row of "States" that is being used within the outer template. This information needs to be specified before the templates can be used to generate pages.



Product Finder

Products & Services > Product Finder

Product List

Category	Model Name	Model #	Interface	Capacity	RPM	Availability
Consumer Electronics Storage	DB35 Series™ 7200.3 SATA 3Gb/s 750-GB Hard Drive	ST3750640SCE	SATA 3.0Gb/s	750GB	7200	
Consumer Electronics Storage	DB35 Series™ 7200.3 SATA 3Gb/s 750-GB Hard Drive	ST3750840SCE	SATA 3.0Gb/s	750GB	7200	
Desktop Storage	Barracuda 7200.10 SATA 3.0Gb/s 750-GB Hard Drive	ST3750640AS	SATA 3.0Gb/s	750GB	7200	

Figure 3.2: Alignment of Fields on a Web Page — On web pages that display tabular data, the alignment of data items as seen on the screen usually reflects the underlying relational structure of the data. Columns of the relational table are aligned parallel to a vertical axis and rows are parallel to a horizontal axis.

tion for all the text elements on all the pages. For each distinct x-coordinate value x_i, the expert builds a list of text nodes that are positioned at x_i. Then, it generates a token hint for all pairs within every text node list.

3.8 Clusters to Relational form

Our approach leaves data in clusters, but it is more useful to have it in relational form. In this section, we present an algorithmic approach to convert clusters to relational tables, which does not guarantee full accuracy in the general case.

Clusters align the values within columns of tables. Finding the relational form is the problem of grouping columns into tables and then aligning the columns into rows such that values in a row belong to a single record.

Let us first examine the problem under some simplifying assumptions: there are no missing values in the relational tables; the list tables contain at least 2 records for every parent record; all the clusters were found with 100% accuracy.

A table is a partitioned set of rows. Each partition of rows contains consecutive sequences of text nodes. A cell can contain a sequence of text nodes or a reference to one of the row partitions of another table. Each column contains either sequences of text nodes

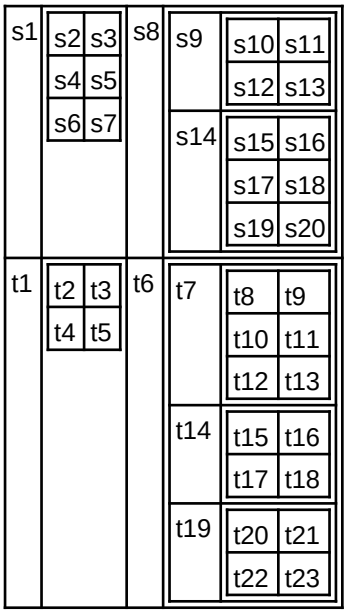


Figure 3.3: Relational Structure of Data — References to values in other tables have been replaced with the actual values. For example, the second column in the first row refers to values stored in another table. These values, [s2,s3],[s4,s5],[s6,s7], are shown as an inner table within a cell of the outer table.

or references to row partitions of one particular table.

We follow a bottom-up approach. First, we form an initial set of columns from the initial set of clusters. Each cluster maps uniquely to a single column. Then we iteratively merge columns into tables and partition the rows of the resulting tables. A set of columns can be grouped as the columns of a table if the columns can be ordered such that for every token in the first column, the next token is in the second column and for every token in the second column, the previous token is in the first column; for every token in the second column, the next token is in the third column and for every token in the third column, the previous token is in the second column; and so on. As an example, suppose that when the clustering approach was run on the site whose relational structure is show in Figure 3.3, it generated the following clusters: c_1={s1, t1}, c_2={s2, s4, s6, t2, t4}, c_3={s3, s5, s7, t3, t5}, c_4={s8, t6}, c_5={s9, s14, t7, t14, t19}, c_6={s10, s12, s15, s17, s19, t8, t10, t12, t15, t17, t20, t22}, and c_7={s11, s13, s16, s18, s20, t9, t11, t13, t16, t18, t21, t23} where s and t are two pages and their contents are the text node sequences (s1, s2, ..., s20) and (t1, t2, ..., t23) respectively. These clusters are also the starting columns which are shown in Figure 3.4 as the leaf nodes. Among these columns, the pair of clusters c_2 and c_3 and the pair of c_6 and c_7

can be grouped as the columns of a table. By taking into account the order of text nodes, we form a partition of the rows of the resulting table. Each partition is a consecutive sequence of tokens and are represented by sequence of text nodes. Thus, the result of merging c_2 and c_3 is a new column of references $c_{2,3}$={[s2-s7], [t2-t5]} and the result of merging c_6 and c_7 is the new column $c_{6,7}$={[s10-s13],[s15-s20],[t8-t13],[t15-t18],[t20-t23]} where [si-sj] denotes the sequence of consecutive tokens between and including si and sj. We then repeat the merging step until no columns can be merged. The sequence of merges is shown in the merge tree. To ensure that tables are not unnecessarily split into multiple tables, the approach follows a strict bottom-up approach. This implies that a set of columns will be merged only if the columns containing adjacent text nodes have been maximally merged and cannot be included in the current merge. For example, c_4 will not be merged with $c_{5,6,7}$={[s9-s20], [t7-t23]} if c_3 has not yet been merged into $c_{2,3}$. Doing this merge without including $c_{2,3}$ would cause the top level table to be broken up into multiple tables.

After all clusters have been merged, finding the relational tables is trivial. Each non-leaf node in the merge tree corresponds to a table. The children of the node are the columns of the table. Each table also has an additional column whose values correspond to the row partitions of the table. For example the table that corresponds to node {[s10-s13],[s15-s20],[t8-t13],[t15-t18],[t20-t23]} is shown below. The partition of rows that represent [s10-s13] is assigned an identifier of 0, [s15-s20] an identifier of 1, and so on.

0	s10	s11
0	s12	s13
1	s15	s16
...	...	...
4	t20	t21
4	t22	t23

When the values in a column are token sequences in the merge tree, these values are represented by references to the partition identifiers of the child table. For example the table that corresponds to node {[s9-s20],[t7-t23]} is the following:

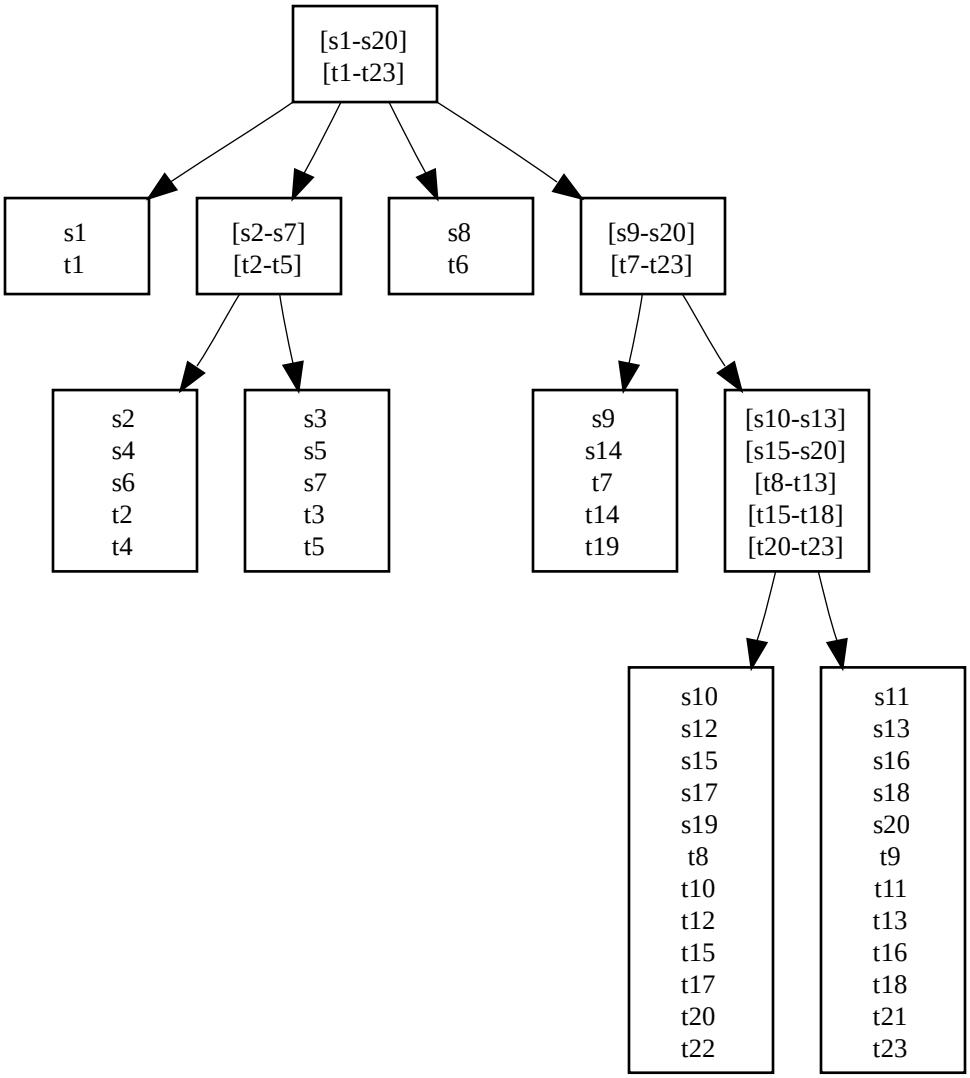


Figure 3.4: Merge Steps to Compose Tables from Clusters — We merge clusters, each of which corresponds to a column, in a bottom-up fashion to form tables.

0	s9	0
0	s14	1
1	t7	2
1	t14	3
1	t19	4

Now let's consider the problem without making the simplifying assumptions we started with. Namely, the relational data contains missing values; lists possibly have less than 2 records; and the clustering step may not find the clusters with 100% accuracy. This affects the approach we described above in several ways. First, the process of merging columns cannot simply rely on matching each token in one column to another token in a second column because for some tokens there will be no matches either because there are missing values or because the clustering step has misplaced some tokens incorrectly. Similarly, the process of determining row partitions cannot rely on finding consecutive sequences of tokens as the clustering step might have incorrectly clustered some of the tokens.

Even though the simplistic approach described above does not solve the problem of finding the relational representation in the general case, the approach can be generalized. In particular, the approach relies on making two types of decisions: merging columns and partitioning rows. In the basic approach above, these decisions are made on strict constraints on the distribution of consecutive tokens into clusters. The problem with these strict constraints is when the clusters are not perfect, the algorithm will not be able to apply merging and partitioning operations. As a first generalization step, the constraints can be softened so that even there is "noise" in the form of missing or incorrectly clustered tokens, the algorithm can continue to apply these operations. For example, the merge step can be made less strict such that rather than requiring matches between all tokens, it can require matches between only a percentage of the tokens. One problem with a less strict approach is that at any point in the process, there might be more than one way to merge columns or partition rows and only one of these might lead to the correct relational structure. Thus, a second level of generalization is to view the problem of finding the relational form from clusters as a search problem where the space of different sequences of merge and partition operations is explored. We leave these types of extensions to our approach as future work.

3.9 Summary

In this chapter, we described a framework for combining heterogeneous experts. A crucial feature of the framework is that it defines a common hypotheses language. This allows arbitrary experts to be combined in a principled fashion. We also described the particular experts we use for the site extraction problem and how we convert clusters, which is an intermediate representation that reflects the underlying structure of the data closely, into relational form.

Chapter 4

Results - CHEX

4.1 Evaluation Methodology

4.1.1 Dataset

Our main goal in evaluating our approach was to demonstrate that it achieves high extraction accuracy in a variety of domains exhibiting differently structured pages. Thus, we collected pages from three types of web sites: E-commerce sites, scientific journals, and company job listings and evaluated our approach against wrappers created using a supervised learning approach as in [46].

The purpose of evaluation is to determine how close the output of the system is to the underlying structure of the data. Thus, evaluation requires both a dataset whose underlying structure is known or can be reliably reconstructed albeit with human help to create an answer key and a measure to determine how well the output matches this structure.

Collecting web datasets whose underlying structure is known is difficult, because this requires access to the data source that the web site uses, but web sites do not normally provide access to their data other than through their web pages.

Given that it is not possible to a build a dataset just by collecting data and its underlying structure from the web, we have two alternative ways to build a dataset. We can either generate synthetic data or start with real web data which we then manually label.

In the first alternative, we can start with a dataset whose structure is known and generate a web site from it, but to do this in a way so that the resulting site is realistic in size and complexity is non-trivial.

Alternatively, we can start with a dataset whose structure is unknown and manually la-

bel the output of the approach. Doing this completely manually is a daunting task because of the vast size of the data, which includes every possible value that can be extracted from the site. Fortunately, the manual labeling effort can be reduced in two independent dimensions. First, we focus only on the main data items on the site, such as fields related to books on a bookstore site, and not other secondary fields, such as advertisements, or presentation fields, such as fonts. Second, the number of actual values that need to be labeled can be reduced by using a semi-automated approach. Thus, we train a supervised extraction system to extract particular data items, validate the output of the supervised extraction system by sampling, and compare the extracted values to the output of our system.

Comparing our approach to other approaches would also be desirable. Unfortunately, site extraction is a new problem and there are no well-established datasets and results that address the general unsupervised extraction challenge. This meant that we were limited to using our own datasets while evaluating our approach in the web domain and that a comparative analysis was not possible. To do a comparative analysis, we decided to build a new set of experts for the record-linkage domain and compare our results with some recently published results. We discuss our work in the record linkage domain in a separate chapter.

4.1.2 Precision/Recall/F1 on Matching Clusters

We have used the following process in evaluating clusterings, which is especially useful when the correct clustering is only partial, i.e., only some of the samples are in the correct clustering: We call the clusters in the correct clustering "target" clusters. We evaluate each target cluster in isolation and then aggregate over the individual results[2]. To do this, for each target cluster, we find the output cluster that contains the largest number of target values. If there is a tie, we pick the cluster with the fewest total values. Then we calculate the retrieved and relevant count (RR), i.e., the number of target values in the output cluster. We also calculate the total number of values in the output cluster (Ret), and the total number of target values (Rel). We then compute precision, recall and F1 scores using the standard definitions of RR/Ret, RR/Rel and $2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall})$, respectively.

As a concrete example, suppose there are three target clusters A, B, and C as shown in Figure 4.1. The evaluation process determines precision and recall values for all the clusters. To determine precision and recall for cluster A, we find the extracted cluster that

contains the most number of values in A, which in this case is cluster 1. All values in cluster 1 are also in cluster A so both the count of retrieved samples (Ret) and retrieved and relevant samples (RR) is 3. This gives a precision score of 3/3 for cluster A. The number of samples in cluster A, which is the count of relevant values, is 5. That gives a recall of 3/5 and F1 of 3/4. Note that cluster A can be evaluated against any cluster, but the evaluation process chooses the cluster that gives the highest recall. Even though the evaluation process picks "good" matches, because each token is clustered into exactly one cluster, the evaluation is still fair. This kind of evaluation would not work if it was possible to have the output contain all possible clusters, but that would require tokens to be in multiple clusters, which is not allowed by our approach. Next, we evaluate precision and recall for cluster B. The extracted cluster that best matches cluster B is cluster 4. This match gives a precision of 5/7, recall of 5/5 and F1 of 5/6. The third cluster C is best evaluated against cluster 4 and that match gives a precision of 2/7, recall of 2/2, F1 of 2/9. Note that both clusters B and C are evaluated against cluster 4. A more strict evaluation method might be not to allow an extracted cluster to be matched to more than one target cluster. In our method, the "penalty" for over-general clusters such as cluster 4 is reflected as a drop in precision, but not in recall, which we found to be satisfactory. Note also that the spurious cluster 2 does not play a role in the evaluation. This in fact is by design and resolves the issue that even though in our approach we extract all of the data, our "labeled" data set is only a small subset of the data. In this particular example, the target clusters do not include any of the members of cluster 2.

4.2 Results

Evaluating our system is challenging because of the size and scope of the problem we address. The output of existing web extraction systems is normally a small subset of the output produced by our system on any given site. For example, if we compare our system against a web-page wrapper, we can only evaluate a few of many clusters because the wrapper would normally extract only a few fields whereas our system would cluster all the tokens on the pages. Manually evaluating the system is also difficult because of the size of the output. So instead of evaluating the full output, we focus here on evaluating how well our system does in extracting target data from different types of web-sites, such as product catalogs, electronic journals and job-listings.

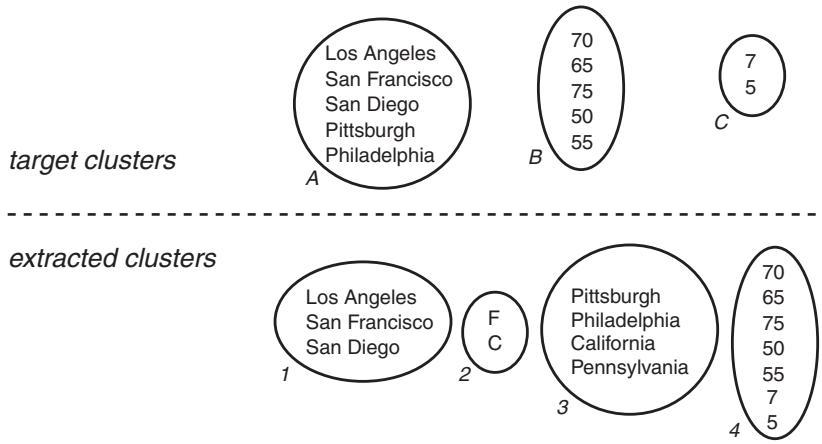


Figure 4.1: Target and Extracted Clusters – To evaluate the accuracy of clustering, we compare the values in each target cluster with the values in each one of the extracted clusters. For each target cluster, we pick the extracted cluster with the highest recall score. Thus, cluster A is evaluated against cluster 1, cluster B against cluster 4 and cluster C also against cluster 4. Note that because each value is placed in exactly one extracted cluster, it is not possible to achieve recall and precision scores of 1.0 simply by generating all possible clusters of values.

Specifically, we compare the output of CHEX to the output of web wrappers that have been created using AgentBuilder [55] (a supervised wrapper induction system) and manually validated for correctness. The output of a wrapper when applied to a set of pages can be represented as a table whose columns correspond to the fields extracted by the wrapper. We refer to these columns as target columns (or, equivalently, target fields) and the extracted data values in each column as the target data.

We report experiments in three domains (Table 4.1). In the first experiment, we compared CHEX to seven wrappers that return data from retail sites. These wrappers were originally built for SRI's CALO project[11] using a supervised extraction system. CALO used these web agents to extract data about products from a specific category of the site's catalog. For example, the agent for buy.com extracts information about projectors for sale.

Ideally, we would like to spider each site, and then run CHEX to get the same data. In practice, the size of these sites – hundreds of thousands of pages[1] – made this impractical. We would end up with too many pages for the current implementation of CHEX to handle (a subject we address later). To scale the size of the problem down, we directed the spider

[1]For example, Google reports 996,000 matches to the query "site:buy.com", which matches any page on buy.com.

Field	RR	Ret.	Rel.	Precision	Recall	F1
Manufacturer	81	81	81	100%	100%	100%
Manufacturer No.	100	100	103	100%	97%	99%
Model No.	66	68	71	97%	93%	95%
Name	97	100	103	97%	94%	96%
Price	77	88	103	88%	75%	81%
Author	1173	1197	1212	98%	97%	97%
Title	1173	1188	1212	99%	97%	98%
URL	528	528	1212	100%	44%	61%
Position	1502	1502	1582	100%	95%	97%
Id	1318	1318	1551	100%	85%	92%
Location	1413	1413	1524	100%	93%	96%

Table 4.1: Summary of Results — We ran experiments in three domains: Journals, E-Commerce Sites, and Job-Listings. We report precision, recall and F1 scores by comparison to data in fields extracted by wrappers induced by a supervised learning system. The output of CHEX includes these fields as well as all other data found on each site.

down to the correct category to collect pages that are relevant to the extraction task. Even with a smaller number of relevant pages, the task is non-trivial as there are multiple types of pages such as list and detail pages. To make the set of pages closer in characteristic to one that a full-spidering would collect, we also spidered the site randomly to collect additional pages. This gave us a collection of pages that is small enough but contains a higher ratio of pages on which the CALO wrappers will work.

The target fields for this experiment were product name, manufacturer, model number, item number (SKU) and price, all of which are generic across product types. We do almost perfectly on 4 of the 5 fields (Table 4.2). On the price field, where we miss some values, we in fact fail only on two of the sites. In both of these sites, CHEX extracts the price as part of a larger field because of variations in the formatting of the price field. Note that our evaluation criteria is rather strict: CHEX gets no credit for the longer field values even though the values include the prices and they are in a cluster of their own. This in turn lowers the overall precision and recall scores in Table 4.2.

Below are sample clusters extracted from overstock.com demonstrating the types of clusters that CHEX extracts from retail sites. The structure of the output mirrors the organization of page and data clusters. Each page cluster contains a set of pages. For each

Web Site			Extracted				
Vendor	Pgs.	Products (Rel)	Mfr (RR/Ret)	Mfg# (RR/Ret)	Mod# (RR/Ret)	Name (RR/Ret)	Price (RR/Ret)
buy	24	12	12/12	12/12	10/12	9/12	0/0
compusa	26	16	16/16	16/16	16/16	16/16	15/16
gateway	22	9	-	6/6	6/6	6/6	6/6
newegg	24	10	10/10	10/10	10/10	10/10	0/10
overstock	26	13	-	13/13	13/13	13/13	13/13
tigerdirect	23	11	11/11	11/11	11/11	11/11	11/11
photoalley	44	32	32/32	32/32	-	32/32	32/32

Table 4.2: Extraction from E-Commerce Sites — All but one field is extracted with high accuracy across a number of sites.

page cluster, the data fields on pages within that cluster are grouped into data clusters. The output displays these data clusters below each page cluster. Looking at the data, it appears that page cluster 9 is a set of list pages and page cluster 11 a set of detail pages. Data cluster 192 is a cluster of category paths. In data cluster 150, CHEX has grouped together some HTML comments, apparently referring to some internal site data. This cluster illustrates the difficulty of evaluating the output of CHEX against supervised wrappers. These comments would not even be visible on a browser, so it is very unlikely that a user would have built a wrapper to extract the data contained in the comments. Data cluster 291 is a cluster of product names cleanly extracted from detail pages. In contrast, cluster 340 contains some HTML markup around extracted data. In fact, the extracted data is the constant "Dimensions" for this set of pages and so not particularly useful. Both of these issues can be solved by post-processing of the data: HTML markup can be stripped off from text and clusters that contain the same value across all the pages can be eliminated from further processing. Data cluster 305 is a cluster of unique identifiers. These types of identifiers are common on retail sites. They are sometimes assigned arbitrarily by the site and sometimes actual product codes. In data cluster 396, CHEX has collected pure HTML with no useful content. Post-processing of the clusters can clearly remove this cluster as well.

```
PageCluster 9 - Pages 7 1 5 11 3 8
  DataCluster 192(6)
   doc48475.HTML Apparel  Shoes & Access. >> Handbags & Accessories
   doc48477.HTML Books  Music  & Videos >> Videos
   doc48473.HTML Electronics & Computers >> Computers & Printers
   doc48474.HTML Electronics & Computers >> Home Office Equipment
```

```
    doc48471.HTML Home & Garden >> Housewares
    doc48476.HTML Worldstock Handcrafted >> World Jewelry
   DataCluster 150(4)
    doc48476.HTML <!- noSkus: 309 -> \n<!- itemCount: 0 -> \n<!- skuPage: 24 ->
    doc48474.HTML <!- noSkus: 39 -> \n<!- itemCount: 0 -> \n<!- skuPage: 24 ->
    doc48471.HTML <!- noSkus: 65 -> \n<!- itemCount: 0 -> \n<!- skuPage: 24 ->
    doc48475.HTML <!- noSkus: 775 -> \n<!- itemCount: 0 -> \n<!- skuPage: 24 ->
PageCluster 11 - Pages 23 18 15 22 14 21 16 20 13 17 24 25 19
  DataCluster 291(13)
   doc48484.HTML Advueu 18-in. LCD Flat Panel Monitor with Speakers
   doc48482.HTML Black 17-in. LCD Flat Panel Monitor with Speakers
   doc48480.HTML Dell 2000FP 20in. LCD Flat Panel Monitor
   doc48488.HTML Dell P1130 Black 21-inch Flat FD Trinitron CRT
   ...
   doc48485.HTML Telart LT17A 17-inch LCD Monitor
  DataCluster 340(13)
   doc48487.HTML <br> \n<b>Dimensions:</b> 
   doc48481.HTML <br> \n<b>Dimensions:</b> 
   ...
   doc48486.HTML <br> \n<b>Dimensions:</b> 
  DataCluster 305(13)
   doc48488.HTML 1042847
   doc48486.HTML 1043475
   ...
   doc48483.HTML 716735
  DataCluster 396(7)
   doc48481.HTML </span><br> \n<table width='100&#37;'><tr><td valign=top> \n<b>
   doc48492.HTML </span><br> \n<table width='100&#37;'><tr><td valign=top> \n<b>
   ...
   doc48488.HTML </span><br> \n<table width='100&#37;'><tr><td valign=top> \n<b>
```

To test CHEX on fully spidered sites, for our second experiment we selected four open-access electronic journals which could be completely spidered: DMTCS, EJC, JAIR and JMLR. We built wrappers for the author, title, and ArticleURL fields for either the "detail pages" (the pages with meta-data on the individual articles) or in the case of EJC for the table-of-contents pages (because it had no detail pages on the individual articles). We ran CHEX on the full collection of pages.

As shown in Table 4.3, CHEX correctly retrieved all the target values on DMTCS and missed only one article on JMLR. CHEX retrieved approximately 90% of the values for JAIR. The missed values are on pages that are clustered separately from the main cluster of detail pages. On the EJC site, there are no individual pages for articles, but all the information is still available from the table-of-contents pages (thus the large difference between the number of pages and the articles). CHEX returned all the target values for the author and title fields, but also included some spurious set of values. For the PDF/PS URL field (for downloading the articles) on the EJC site, the results show 0 retrieved values because CHEX returned a longer field containing multiple links to different formats of the article, e.g.:

```
<A HREF="PostScriptfiles/v11i2r9.ps">ps</A> | <A HREF="PDF/v11i2r9.pdf">pdf</A>
```

For the last experiment, we picked 50 companies web sites with online job listings

Web Site			Extracted		
Journal	Pages	Articles (Rel)	Author (RR/Ret)	Title (RR/Ret)	URL (RR/Ret)
DMTCS	128	112	112/112	112/112	112/112
EJC	19	645	645/669	645/660	0/0
JAIR	347	297	259/259	259/259	259/259
JMLR	183	158	157/157	157/157	157/157

Table 4.3: Extraction from Electronic Journals — CHEX extracts almost all fields with 100% accuracy.

from the Forbes 500 list. On each of these sites, we spidered from the main job-listings page down to the individual posting pages. Among the sites, the common fields were position, requisition-id, and location, so we decided to evaluate CHEX on those. In Table 4.4, we report the results for 41 of the 50 sites. CHEX extracts 73% of the fields with precision and recall scores higher than 0.9. On the fields where CHEX fails, the failure is usually because CHEX extracts a larger field containing the target field.

The remaining 9 sites out of the 50 proved to be too difficult for CHEX. The set of experts we used in the current version were not able to find structures correctly. The most common cause of this was that some sites include long job descriptions in free-form text with little HTML markup. Not only the experts have no particular understanding of text, but they attempt to interpret text by breaking it into tokens This generates short and usually meaningless segments with the result that the actual structures that the experts look for become much harder to find. For example, the template structure easily gets obscured by the long sequence of words in natural language text. In domains where free-form text can be treated as atomic for the purposes of extraction, a pre-processing step where long sequences of text is replaced with place-holder tokens would be useful so that individual experts don't have to be concerned with free-form text. In the more sophisticated approach we describe in the next chapter, we address this issue by working on DOM text nodes which provide a more meaningful decomposition of long free-form text segments.

Note that in the experiments, we report mainly on fields of base tables, which have only one value per page, and not on fields of list tables, which may have any number of values on a single page. List fields are generally more difficult to extract and the results are often harder to evaluate.

Web Site			Extracted(RR/Ret)		
	Pgs.	Jobs(Rel)	Position	ID	Location
altera	49	13	13/13	13/13	13/13
amer. tower	44	32	32/32	32/32	32/32
amerus	51	31	31/31	31/31	31/31
assoc. bank	39	16	16/16	0/0	16/16
avalonbay	29	25	25/25	25/25	25/25
bankunited	63	20	20/20	20/20	20/20
bea	114	93	47/47	47/47	47/47
broadcom	22	10	10/10	10/10	10/10
carolinafirst	253	84	84/84	84/84	84/84
devon	96	58	40/40	40/40	40/40
devonenergy	91	53	52/52	52/52	52/52
ea	28	15	15/15	15/15	15/15
eogresources	42	20	20/20	20/20	20/20
equitable	28	8	8/8	8/8	NA
fbr	33	30	30/30	30/30	30/30
flagstar	159	141	136/136	136/136	136/136
indymac	106	101	100/100	100/100	100/100
insight	63	31	31/31	NA	31/31
juniper	49	21	19/19	19/19	19/19
markel	33	19	16/16	16/16	16/16
medimmune	147	115	115/115	115/115	115/115
microchip	47	37	37/37	37/37	37/37
mylan	36	26	26/26	26/26	26/26
ncen	133	132	132/132	132/132	132/132
oge	11	5	5/5	5/5	5/5
oldnational	72	71	71/71	0/71	71/71
patterson	26	4	4/4	4/4	4/4
pepco	15	10	10/10	10/10	10/10
phoenix	23	15	15/15	15/15	15/15
pixar	32	27	23/23	23/23	23/23
pnc	45	20	20/20	20/20	20/20
protective	37	31	31/31	31/31	0/31
qlogic	61	59	59/59	59/59	59/59
rga	57	16	16/16	0/0	16/16
simon	66	49	49/49	49/49	49/49
skyfi	39	10	10/10	10/10	10/10
tollbrothers	53	50	50/50	50/50	50/50
troweprice	59	50	50/50	0/0	NA
trz	39	14	14/14	14/14	14/14
whitney	59	10	10/10	10/10	10/10
wilm. trust	14	10	10/10	10/10	10/10

Table 4.4: Extraction from Sites with Job Listings — On this domain, we evaluated CHEX on a larger dataset.

4.3 Observations

In the approach we described in the this chapter, experts make binary decisions in that for a given pair, an expert chooses to output the hypothesis to state that the pair is in the same cluster or chooses to *not* output the same hypothesis. When an expert does not generate the hypothesis for a pair, it is ambiguous whether the expert has no knowledge about the pair or it considers the pair unlikely to be in the same cluster. This is a shortcoming of the approach because we'd like to allow the experts to be able to express hypotheses that samples should not be in the same cluster.

Another shortcoming of the approach is that for a large class of experts the decision to generate hypotheses is intrinsically not binary. Any expert that computes a distance or similarity measure between samples falls into this class. An obvious way to turn the range of values into a binary decision is by thresholding but this potentially loses information that may be useful while combining the hypotheses.

An interesting property of our approach is that unlike standard clustering approaches where the clustering granularity is specified by an external parameter such as number of clusters or intra-cluster distance threshold, the probabilistic evaluation naturally finds the optimal granularity. This also means that the trivial solution of a single cluster of all samples, which by definition satisfies any set of hints, is not necessarily the optimal solution. A clustering where clusters include only a minimal set of samples to satisfy the given set of hypotheses generally leads to a higher conditional probability of the given set of hypotheses and thus is the optimal solution.

Chapter 5

CONFHEX: Clustering with Confidence Scores from Heterogeneous Experts

5.1 Overview

In the previous two chapters, we defined a basic framework for combining heterogeneous experts, applied it to data extraction in the web domain, and reported on the results of our experiments with it. Although the basic framework performs well, it has some some shortcomings, namely that experts cannot hypothesize that samples are *not* in the same cluster and that any level of confidence an expert may assign to its hypotheses is not available at the time when hypotheses are combined. In this chapter, we address these shortcomings by extending the hypothesis language so that experts assign confidence scores to the hypotheses they generate, but leave other "shortcomings" such as coping with unstructured text for future work.

A confidence score on a hypothesis represents a probability estimate of that expert that the statement is going to be true in the solution. A score of 0 indicates that the expert believes the hypothesis to be false and a score of 1 indicates that it believes the hypothesis to be true. If a hypothesis is completely out of the domain of an expert, it can assign the hypothesis a score of 0.5 indicating that as far as the expert is concerned, the hypothesis is equally likely to be true or false.

5.2 Bayesian Network

5.2.1 Structure

In CHEX, we defined an optimal clustering as one that maximized the probability of the hypotheses given the clustering. To compute this probability, we defined a stochastic process of hypotheses generation from clusterings. This kind of approach assigns a probability to a given set of hypotheses for a given clustering, but cannot take into account confidence scores assigned to the hypotheses, which is what we would like to do in CONFHEX.

Instead of attempting to define a stochastic process of hypotheses generation as we did in CHEX, here we utilize Bayesian networks, which have been extensively studied for representing hypotheses and confidence scores. We use the conceptual structure of a Bayesian network and represent hypotheses with nodes in the network without actually using the network to compute probabilities.

In typical applications of Bayesian networks[44, 74], there are a number of variables which model the hidden state of the world and another set of variables which represent the observations. The structure and parameters of the network represent the dependencies between all the variables, but most importantly between hidden variables and observed variables, and allows the distribution of assignments to the hidden variables to be determined given the assignments to the observed variables.

As in CHEX, our goal is to find the optimal clustering given a set of hypotheses. We represent the clustering problem as a Bayesian network as follows (Figure 5.1. The structure of the network is a tree. There is one root variable C which represents the clustering and whose domain is the set of all clusterings (of the given samples). The immediate descendants of this node represent the set of hypotheses on pairs. Let's call these variables L_{ij} where i and j range over the samples [L stands for link]. The domain of L_{ij} is true, false. The descendants of each L_{ij} are nodes that represent experts and there is one such node E_{kij} for each expert where k ranges over the set of experts.

These nodes are *virtual* because rather than being true random variables that range over the output of the experts, they always take on the "observed" value together with a confidence score assignment[58]. In contrast, a true observed variable would have a proper domain and assigned a single value from that domain (with the implied confidence score of "1").

The Bayesian network determines the full joint distribution of variables C, L_{ij}, and

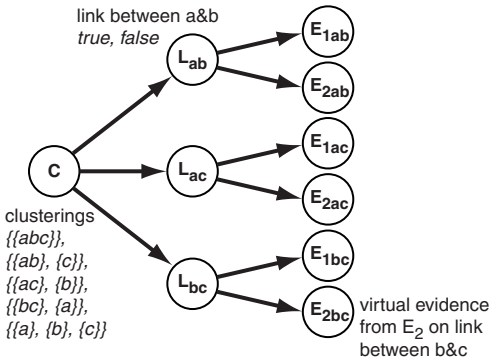


Figure 5.1: Bayesian Network for Clustering — Evidence as collected by the experts enter the network at the leaf nodes and determines the probability of a pair of samples being in the same cluster. This is then used to find the most likely assignment to the root node C which ranges over all possible clusterings of the samples.

E_{kij}. Our goal is to find the most likely value of the clustering node C given the values of E_{kij}, which represent the outputs from the experts. In doing this we are interested in computing the values of $P(C = c|Evidence)$ for different values of c, where $Evidence$ is the set set of E_{kij}. First, we apply the Bayesian theorem:

$$P(C = c|Evidence) = \frac{P(C = c \wedge Evidence)}{P(Evidence)} = \frac{P(Evidence|C = c)P(C = c)}{P(Evidence)}$$

The factor $P(C = c)$ is a constant because we assume that all clusterings are equally likely. It does not need to be taking into account in determining the most likely value of C. The prior probability of evidence $P(Evidence)$ does not depend on c and so does not play a role in determining the most likely value of C either. The last factor $P(Evidence|C = c)$ can be computed with the assumption that the experts' outputs, E_{kij}, are independent given a clustering:

$$P(Evidence|C = c) = \prod_{i,j,k} P(E_{kij}|C = c)$$

In computing the factors of this product, the intermediate nodes L_{ij} need to be taken into account. The domain of L_{ij} is $True, False$, so:

$$\begin{aligned} P(E_{kij}|C = c) &= P(E_{kij}|L_{ij} = True)P(L_{ij} = True|C = c) \\ &\quad + P(E_{kij}|L_{ij} = False)P(L_{ij} = False|C = c) \end{aligned}$$

Of the two terms, one will always be zero. That's because $P(L_{ij} = True|C = c)$ is 1

P(C)

c	P(C=c)
abc	0.2
ab,c	0.2
ac,b	0.2
bc,a	0.2
a,b,c	0.2

$P(L_{ab}|C)$

c	$P(L_{ab}$=True|C)	$P(L_{ab}$=False|C)
abc	1.0	0.0
ab,c	1.0	0.0
ac,b	0.0	1.0
bc,a	0.0	1.0
a,b,c	0.0	1.0

$P(E_{1ab}|L_{ab})$

l_{ab}	$P(E_{1ab}$=Observed|$L_{ab})$
True	0.7
False	0.3

Figure 5.2: Sample Probability Tables — The probability table for the root node C reflects that all clusterings are equally likely, the probability table for L_{ab} reflects the deterministic relation between a particular clustering and the existence of a link between samples a and b, and the probability table for E_{1ab} reflects the confidence of expert E_1 on the hypothesis that a and b are in the same cluster.

when samples i and j are in the same cluster in c and 0 when they are not. Thus:

$$P(E_{kij}|C=c) = \begin{cases} P(E_{kij}|L_{ij}=True) & \text{if } i,j \text{ are in the same cluster in } c \\ P(E_{kij}|L_{ij}=False) & \text{otherwise} \end{cases}$$

The last equation leads to a method to compute unnormalized probabilities for clusterings. To determine the conditional probability of a clustering c given the output of the experts, first determine whether each L_{ij} is true or false by checking if samples i and j are in the same cluster in c. Then for each expert E_{kij}, multiply the probabilities $P(E_{kij}|L_{ij})$. The product of all these intermediate factors is the unnormalized conditional probability of clustering c.

5.2.2 Parameters

Corresponding to the three levels of the Bayesian Network, there are three types of probability tables. Here we describe how the probabilities in each of these tables are determined.

The first is the probability distribution of variable C. As before, we assume each clustering is equally likely and assign a uniform distribution to C. The second is the conditional probability table for L_{ij}. This table contains values for $P(L_{ij}|C)$. Given a particular clustering, the value of L_{ij} is fully determined, thus the values are either 0 or 1. If the samples i and j are in the same cluster in C, then $P(L_{ij}|C)$ is 1. Otherwise, it is 0. The third type of parameter, $P(E_{kij}|L_{ij})$, represents the belief of expert k in the hypothesis L_{ij}.

Figure 5.2 shows the probability tables for a clustering problem of three samples a, b, and c. The table for $P(C)$ assigns a uniform probability of 0.2 to each clustering as there are a total of five clusterings of 3 samples. The table for $P(L_{ab}|C)$ demonstrates the deterministic relation between the clustering variable C and the hypotheses L_{ij}. For example when C takes on the value ac, b, the value of L_{ab} is $false$ by definition. Thus, $P(L_{ab} = True|C = ac, b) = 0$ and $P(L_{ab} = False|C = ac, b) = 1$. The third table, $P(E_{1ab}|L_{ab})$, contains confidence scores assigned to the hypothesis L_{ab} by expert 1. In this particular example, expert 1 has assigned a score of 0.7 to $L_{ab} = True$ and 0.3 to $L_{ab} = False$. The next sections describe what these numbers mean and how they are computed.

5.2.3 Confidence Scores

When an expert generates output, it assigns a confidence score to each hypothesis. One question is how experts arrive at these confidence scores. We discuss two alternatives: each expert determining its own confidence score and the Bayesian network modeling the output of each expert.

Virtual Nodes

One approach is to make it the responsibility of the expert to determine the confidence scores which are introduced to the Bayesian network via the virtual nodes E_{kij}. This provides some flexibility in that each expert can use a different method to determine confidence scores. For example, experts that are hand-coded to represent some human expert knowledge can use the human expert's subjective confidence level. Experts that rely on "global" statistics (e.g., distribution of first names) can use these global statistics to assign confidence scores.

Leaving aside the subjective confidence level, the mapping of experts output to confidence scores can be viewed as a learning problem. More specifically, the problem is to estimate $P(E_{kij}|L_{ij})$ where L_{ij} is either true or false. When L_{ij} is true $P(E_{kij}|L_{ij})$ is the distribution of the output that the expert generates given that the samples are in the same cluster. We call this the *within-cluster* distribution. For example, a string-edit distance expert would generate output that is skewed towards smaller values when the samples are in the same cluster (i.e., refer to the same entity). The second distribution, $P(E_{kij}|L_{ij})$ when L_{ij} is false, is the distribution of the output when the samples are not in the same clus-

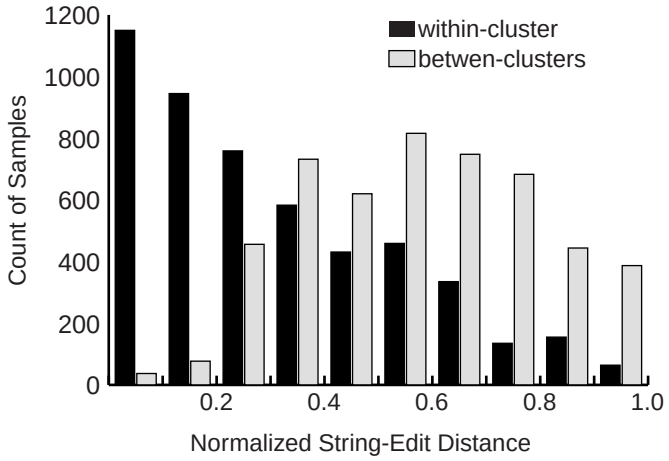


Figure 5.3: Distribution of Distance Scores for an Expert — For this particular expert, the distribution of distance scores is skewed towards small values for pairs that are in the same cluster and towards larger values for those that are in separate clusters.

ter. We call this distribution, the *between-cluster* distribution. In Figure 5.3, the within-cluster and between-cluster distributions of sample counts for a string-edit based expert are shown. To determine these distributions, we used training data in which samples were clustered manually. We ran the string-edit expert on the data, which assigned a distance to each sample pair. The distances were normalized by the total length of the strings so that the values fall between 0.0 and 1.0. We next divided the sample pairs into two sets to determine the within-cluster and between-cluster distributions. One set contained pairs such that both samples are in the same cluster and the other set contained the remaining pairs. Then, we counted the number of sample pairs whose distance fall into bins of width 0.1 for both sets and plotted the graph shown in Figure 5.3. As expected the within-cluster distribution is clearly skewed towards smaller values and the between-cluster distribution to relatively larger values.

If the form of the distributions can be determined, learning the mapping from expert output to confidence scores reduces to estimating the parameters of these distributions. For example, if the expert generates output that tend to fall into a normal distribution for samples from the same cluster, then the statistics collected from the training data determines $P(E_{kij}|L_{ij} = true)$. Alternatively, the output can be discretized into the buckets of a multinomial distribution to approximate the unknown form of the actual distribution. The statistics collected from the training data determine the probabilities of the buckets.

The statistics we collect from training data is potentially noisy and can lead to proba-

bility values of zero. To reduce some of this noise, we apply a smoothing technique to the multinomial distribution. In particular, we smooth our initial estimate of the distribution by computing its convolution with a Gaussian. The smoothing step flattens the probability distribution such that probabilities which are near the extremes of 0 and 1 are pulled towards 0.5. The amount of smoothing depends on the standard deviation of the Gaussian. We determined 0.05[1] to work sufficiently well empirically and have consistently used it to smooth the distribution estimates.

After training, an expert goes through the following steps for each sample pair i, j:

1. Generate output for the given sample.
2. Map the discretized output value to a confidence score.
3. Set E_{kij} to "observed" in the Bayesian network and $P(E_{kij}|L_{ij} = true)$ to the confidence score.

Observed Nodes

A second and simpler approach is to model the multinomial distribution within the Bayesian network directly. To do this, we extend the domain of E_{kij} to all the discretized values of expert k. Then the conditional probability table of $P(E_{kij}|L_{ij})$ in the Bayesian network directly represents the multinomial distribution. The values of this table can be determined using the same approach as above, but now the expert does not need to map its output to a confidence score as that step becomes the responsibility of the network. Thus, the steps are:

1. Generate output for the given sample.
2. Set E_{kij} to the discretized output value.

None of the experts we have used for our experiments has an intrinsic mechanism to assign confidence levels. Each expert simply outputs similarity or distance scores, which the second approach allows to be directly introduced to the Bayesian network. Thus, we use observed nodes in our Bayesian network model.

5.2.4 Belief Propagation

In a typical application of a Bayesian network, once the values of the observed variables are assigned, the belief is propagated to the hidden variables and thus the distribution of the

[1] All scores are normalized so that they take on values between 0 and 1 and discretized into 10 bins.

unknown state of the world is discovered. The propagation of belief through the network can be done with efficient algorithms.

Unfortunately, standard propagation algorithms cannot be efficiently applied to the Bayesian network described here. This is because in the typical usage, the hidden state of the world is represented by a number of variables each of which has a small domain, but in our representation, it is represented by a single variable with a very large domain. The standard propagation algorithms ultimately find the distribution of each of the variables and so compute the probability of every value in the exponentially large domain of the clustering variable. This takes an exponentially long time.

The difference between the typical Bayesian model and ours arises from inherent consistency constraints of the clustering domain. An alternative representation is to remove the root variable C from the network. The remaining network can be evaluated efficiently as the only remaining hidden variables are L_{ij} whose domains are true, false. Unfortunately, it is not possible to use the independent distributions of L_{ij}. The most likely value of some node might not be the most likely value of that node if all the other nodes are taken into account, because the most likely values of other nodes might be in conflict with the most likely value of that node.

As before, we are interested only in the most likely value of the clustering node and not in its full distribution. Thus, we revert to a search-based approach to find the most likely clustering and use the Bayesian network to evaluate particular clusterings.

In evaluating a particular clustering c, the network assigns $P(C = c|Evidence)$ the product of all P_{ij} where $P_{ij} = P(L_{ij} = True|Evidence)$ if s_i and s_j are in the same cluster in c and $P(L_{ij} = False|Evidence)$ otherwise. Each $P(L_{ij}|Evidence)$ can be computed by simply propagating belief from its children, namely E_{kij}.

5.3 Combining Experts

The approach combines expert in two ways: first, in assigning the combined confidence score to a single hypothesis that a particular pair is in the same cluster, and second, in finding the most likely clustering given all the hypotheses.

Combining the confidence scores of multiple experts for individual hypotheses using the Bayesian network is equivalent to naive-Bayes learning for binary classification. Consider the problem of determining if a pair of samples is in the same cluster given a set

of similarity scores from multiple experts. This problem is a binary classification problem on pairs where features are defined on pairs (rather than on individual samples) in the form of experts. Each feature represents a different method of computing a similarity measure on a pair of samples. The classification problem is to assign a value of true or false to a pair based on the values of these features. This can be cast to a probabilistic framework as finding the value of H such that $P(H|F_1, F_2, ..., F_n)$ is maximized, where H is the hypothesis that a particular pair is in the same cluster and F_i are the n experts. We can simplify this problem by assuming that the similarity score from each expert is conditionally independent give the hypothesis. This assumption leads to the well-known formula:

$$P(H|F_1, F_2, ..., F_n) = \frac{P(H)\prod_i P(F_i|H)}{Z}$$

where Z is a constant representing the prior probability of observing the features. This formula is essentially identical to the one used in propagating belief from the expert nodes to L_{ij}. This is expected as the structure of the Bayesian network makes the naive-Bayes assumption explicit: The only dependence relation between an expert and the rest of the graph is through the hypothesis node L_{ij}. Thus, knowing the value of L_{ij}, which corresponds to H above, breaks the transfer of information from child expert nodes E_{kij}. In other words, experts are conditionally independent given the hypothesis.

The more interesting case is combining the confidence scores on multiple hypotheses to find a consistent model. The approach we follow here enforces consistency by restricting solutions to only consistent subsets of hypotheses. These subsets of consistent hypotheses is defined by the domain of root node which varies over clusterings and by the conditional probability tables of L_{ij} which set the probability of inconsistent hypotheses to zero.

The following example illustrates how consistency is maintained by the root node and how this affects the distributions of various variables through propagation. First, we'll use a simplified network from which E_{kij} have been removed. Thus, the observations will be whether a hypothesis is true or not instead of the output of experts. The network has 4 nodes for clustering 3 samples a, b, and c: one root node C whose domain is the set of five clusterings $\{a, b, c\}$, $\{a, bc\}$, $\{b, ac\}$, $\{c, ab\}$, and $\{abc\}$; and three nodes L_{ab}, L_{ac}, and L_{bc} whose domains are true, false. We assume that each clustering is equally likely so the distribution of C without any observations is the uniform distribution: P(C=c)=0.2 for clustering c. Suppose now hypothesis L_{ab} is observed to be true. Propagating the

new distribution of L_{ab} through its conditional probability table to node C gives the new distribution for C: $P(C = c|L_{ab} = true) = .5$ for c=c, ab or c=abc and 0 otherwise. At this point, the clusterings that are not in agreement with L_{ab} have 0 probability. Suppose next hypothesis L_{ac} is observed to be true. Propagating this distribution to node C updates the distribution of C so that the only possible value of C is abc, the only clustering that is in agreement with both L_{ab} and L_{ac}. Note that the conditional distribution of L_{bc}, the remaining hypothesis, reflects that its value is determined by the observations on the other hypotheses: $P(L_{bc} = true|L_{ab} = true, L_{ac} = true) = 1$. Clearly, assigning false to L_{bc} as the observed value results in an inconsistent set of observations and the probability of any clustering becomes 0.

Propagation of belief when the observations are on outputs of the experts follows the same pattern, but inconsistent beliefs are handled gracefully. Suppose the combined belief from the experts is 0.7 on L_{ab} and 0.8 on L_{ac}. This results in the following distribution for node C:

$P(a, b, c|L_{ab} = true, L_{ac} = true) = 0.06$
$P(a, bc|L_{ab} = true, L_{ac} = true) = 0.06$
$P(b, ac|L_{ab} = true, L_{ac} = true) = 0.23$
$P(c, ab|L_{ab} = true, L_{ac} = true) = 0.13$
$P(abc|L_{ab} = true, L_{ac} = true) = 0.53$

Suppose now the combined belief on L_{bc} is 0.4. In a non-probabilistic setting where confidence scores below 0.5 imply negation, a confidence score of 0.4 on L_{bc} would mean that b and c are not in the same cluster, but this is in conflict with the previous hypotheses which state that samples a and b are in the same cluster as well as samples a and c. Fortunately, the conflicting belief can be propagated through the network as before to compute the new distribution of P.

$P(a, b, c|L_{ab} = true, L_{ac} = true) = 0.07$
$P(a, bc|L_{ab} = true, L_{ac} = true) = 0.05$
$P(b, ac|L_{ab} = true, L_{ac} = true) = 0.28$
$P(c, ab|L_{ab} = true, L_{ac} = true) = 0.16$
$P(abc|L_{ab} = true, L_{ac} = true) = 0.44$

The hypothesis that samples b and c should not be in the same cluster reduce the posterior probability of $\{abc\}$ and $\{a, bc\}$ as would be expected intuitively, but the level of confidence in L_{bc} is not high enough to change the most likely clustering. In this particular case, the belief in L_{bc} needs to be less than 0.3 before the most likely clustering switches

to $\{b, ac\}$ from $\{abc\}$.

5.4 Web Domain

5.4.1 Overview

In the web domain, we solve two clustering problems: page clustering and data clustering. These problems are dependent in that solving the page clustering problem makes the data clustering easier and vice versa, so a co-clustering approach as in [26] or [25] is reasonable. However, we were able to achieve good results without co-clustering techniques in both the web domain and the record linkage domain. We leave it as future work to determine whether further improvement in performance is possible by applying co-clustering on top of our framework.

In page clustering, the goal is to find clusters of pages such that each cluster corresponds to one page-type. In data clustering, the goal is to find clusters tokens such that each cluster corresponds to one column of a relational table.

5.4.2 Page Clustering

Being able to cluster pages according to their page-type, a set of pages that contain the same type of data and have similar HTML structure, is a useful step in unsupervised data extraction. If an accurate page clustering is found, then data clustering, which we describe in the next section, can take advantage of the page clustering. One approach to doing this is by limiting data clustering to within page clusters, which reduces the overall size of the data clustering problem by dividing it into multiple independent problems.

To cluster web pages from a web-site, searching multiple types of structure is useful, if not necessary. As in the previous chapters, our approach to page-clustering is to build experts for finding different types of structures. Each expert focuses on a particular structure and passes its discoveries as hypotheses into the Bayesian network. Unlike CHEX, the Bayesian network approach allows experts to assign confidence scores to hypotheses. This in turn expands the types of experts that can be combined in CONFHEX.

We use the following experts for page clustering:

Experts

- **URL Expert**: As in CHEX, this expert takes advantage of the structure of the URLs on a given site. Two pages that are of the same type will normally have similar URLs. In contrast to CHEX where the expert uses a threshold on similarity scores, here the similarity scores are introduced to the Bayesian network as evidence. The similarity of the URLs of two pages is computed from the length of the longest common subsequence of their characters.
- **Template Expert**: We have defined templates and one type of template expert in Chapter 3. In CONFHEX, we use the template structure to determine a similarity score on a pair of pages. The template expert determines the similarity of two pages by comparing the length of the template to the length of the pages. The longer the sequence, the more likely the pages are to be in the same cluster.
- **Layout Expert**: As in CHEX, we make use of the layout structure of pages. In page clustering, the layout experts compute a similarity score from the layout of a pair of page as follows: The layout expert first computes the distribution of DOM nodes along the x-axis. This is simply the number of DOM nodes with the same x-coordinate at each position along the x-axis. The expert then assigns a similarity to a pair of pages by comparing the distributions of the two pages. In particular, the expert casts the distributions as vectors (where each position is a dimension) and finds the cosine similarity between the two "vectors" of the documents. The idea behind this expert is that similar pages, especially similar list pages, have many items that are all positioned at the same x-coordinate and their DOM node distribution along the x-axis will be similar.
- **List Expert**: We use the same expert as in CHEX to analyze list structure, but instead of generating token hints, we use the list structure to compute a similarity score between two pages based on the coverage of the template.
- **Table Expert**: This is a new expert that we introduce in CONFHEX. It uses template coverage to determine similarity but only after finding HTML tags that are commonly used to represent tabular data and then dropping all but the first few rows of each table. The idea behind this expert is that even when tables (and lists) have widely varying number of elements across multiple pages, if the pages are of the same page-type, then the first few rows, which typically include column names,

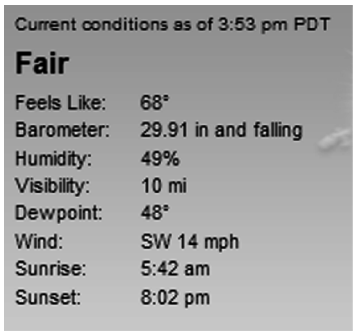


Figure 5.4: Sample HTML as Displayed in a Browser — Only text nodes, such as "Feels Like:" and "SW 14mph" are visible on screen.

from corresponding tables will be similar.

5.4.3 Data Clustering

To automatically extract data from web pages, we cluster text segments so that each cluster corresponds to a single column of a relational table. We can then determine the relational tables by examining the clusters of text segments and the order in which they appear on the web pages.

One problem is determining the text segment boundaries. We bypass this problem by taking advantage of the HTML structure of pages. The HTML parse tree of a web page already separates the text into reasonably-sized segments: Any sequence of characters that are not part of HTML markup becomes a single text segment. Thus, given a web site, we first determine the set of text segments using the HTML parse tree and then run our clustering algorithm on the set of text segments. As an example, take the following HTML snippet which is taken from the web page shown in Figure 5.4:

```
<div class="forecast-module">
<em>Current conditions as of 3:53 pm PDT</em>
<h3>Fair</h3>
<dl>
<dt>Feels Like:</dt><dd>68&deg;</dd>
<dt>Barometer:</dt><dd>29.91 in and falling</dd>
...
```

The parse tree for this segment is as follows:

```
Node: div
  Node: em
    Text: Current conditions as of 3:53 pm PDT
  Node: h3
    Text: Fair
  Node: dl
    Node: dt
      Text: Feels Like:
    Node: dd
      Text: 68&deg;
    Node: dt
      Text: Barometer:
    Node: dd
      Text: 29.91 in and falling
...
```

The set of text nodes in the parse tree define the set of samples for clustering. In this example, the sample set would include the 6 text segments "Current conditions as of 3:53 pm PDT", "Fair", "Feels Like:", "68°", "Barometer:" and "29.91 in and falling". Note that even though the samples are just text segments, the experts can still use contextual information. For example, we have an expert that determines the similarity of two text segments by comparing the path from each node to the root of the HTML parse tree. This expert clearly uses information that is only available when the text segment is considered within its HTML context.

Experts

We group the data experts into three types. The first type of expert uses only the content of the text segment in assigning similarity scores. In other words, a pair of text segments will be assigned the same similarity score regardless of where on a page and on a web site they appear. The second type uses context from the page. This implies that a pair of text segments may be assigned a different similarity score depending on where they appear on the page. In fact, our current experts of this type rely only on context and not content in assigning similarity scores. The third type of expert also uses contextual information, but the context is global in the sense that it can extend to all spidered pages and links. The template experts are of this third type in that the similarity score depends on a template which is induced from multiple pages.

The following experts are content-based:

- **Date Expert**: Determines the similarity of two text segments by comparing the dates they contain. The date expert can interpret many different representations of

dates. To do this, it searches for a variety of patterns so that it can find dates such as “2/16/07”, “20 Jan 2007”, and “May 23”. It assigns a similarity score to a pair of samples based on the formats of the dates it finds in each sample. Two samples that contain the same format are assigned a larger similarity score than two samples that contain dates in different formats which are in turned assigned a larger similarity score than two samples only one of which contains a date. The expert computes the similarity score by first transforming each sample into a sequence of pattern identifiers, then finding the string-edit distance between the two pattern sequences and finally normalizing the distance by the length of the sequences.

- **Levenshtein Expert**: Determines the similarity of two text segments by calculating the Levenshtein distance between the character sequences of the two. The distance is normalized by the sum of lengths of the text segments.
- **Text Pattern Expert**: Determines the similarity of text segments by analyzing the sequence of character classes of the two. For example, ”12.49” and ”109.49” are similar because both contain a sequence of digits, followed by a period, followed by a sequence of digits. This is done by tokenizing each text segment and then determining the similarity of the token sequences with Levenshtein distance. The goal of tokenization is to reduce character sequences into more generic tokens. In our current implementation, we have four such generic tokens: <consecutive upper case letters>, <consecutive lower case letters>, <digits>, and <spaces>. All other characters are represented as individual tokens and maintain their identity. In the example above, both sequences are tokenized as <digits><.><digits>. Thus, the distance between them is 0.

The next two experts use page context:

- **DOM Path Expert**: Analyzes the HTML structure of pages and determines the similarity of two text segments by comparing their paths in the HTML parse tree. The path of a text segment is the sequence of ancestor nodes up to the root of the tree. It is common for web sites to use the same kind formatting to display particular types of information even when the information appears in different context on different pages. The idea behind this expert is to identify text segments that are surrounded by similar HTML tags even though the segments may appear in different parts of the tree. For example, we would like the expert to assign a high similarity score to two bold segments that are items in a list which is itself in a table cell. To do

this, the expert first determines the path from each text node to the root of the parse tree. In this particular example, the path, going from immediate parents to the root, start as follows: <B> <LI> <UL> <TD> <TR> <TABLE> ... The expert then finds the first location on the two paths that don't refer to the same type node. In the example, the paths are identical all the way to the 6th position which refers to the <TABLE> node. If the next elements are not the same, then the first position where paths differ would be the 7th. The expert then normalizes this number by the total length of the paths so that identical paths get a similarity score of 1 and paths that differ in the first position get a similarity score of 0.

- **Layout Expert**: Analyzes the layout of pages and hypothesizes that two text segments are in the same cluster if they are aligned in columns, that is, they have the same x-coordinate on the screen. Most lists on web pages are laid out so that items in the list are precisely at the same x-coordinate. The same observation is true for columns of tables. This expert first determines the x-coordinate of text nodes by loading pages into a browser window and querying the browser. It then assigns a similarity of zero to pairs of text nodes that have the same x-coordinate and one to pairs that do not.

The next two experts use global context:

- **Template Expert**: Analyzes the token sequence of pages to find unique sequences of tokens that are common to all the pages. These common sequences determine a template for the given set of pages. This expert hypothesizes that text segments that fall into the same slot of a template are in the same cluster.
- **DOM Template Expert**: Analyzes the HTML structure of a page to find repeating patterns and hypothesizes that segments that match the same "slot" of a pattern are in the same cluster. The expert first applies the Hierarchical Template algorithm, described in detail in an earlier chapter, to each page. This results in a template which contains sub-templates each of which matches repeating subtree structures within the HTML parse tree. For example, a page that contains a list of product matches to a query as well as a list of product categories will result in a template that has two sub-templates: one sub-template will match the rows of the product list and another sub-template will match the rows of the category list. Applying the template to the page gives the data segments that fall into the slots of the sub-templates. These slots usually correspond to the fields of the items in each list. For example, the slots

of first sub-template above might be the product title, detail URL, and the price and the slots of the second might be the category name, and the category URL. Each slot can be identified by its position in the hierarchy of templates. For example, the first slot of the second list is different from the second slot of the same list and different from the first slot of the first list. The DOM template expert assigns a similarity score to text segment pairs in relation to the hierarchical distance between the slots in which they fall. Text segments that fall into the exact same slot, i.e., in the same column of the same list, get the highest similarity score. Segments that are in different top-level lists get the lowest score.

5.5 Summary

In this chapter, we developed a second framework for clustering with heterogeneous experts. In the new framework, experts assign confidence scores to their hypotheses. This allows more information from the experts to be available at the time when the hypotheses are combined.

In the next chapters, we first describe our experiments in the web domain and report our results. Then, we apply the same framework to the record linkage domain and do a comparative analysis on our results.

Chapter 6

Results - CONFHEX

6.1 Goals

We have several goals in evaluating our approach. Our main goals are to demonstrate that our approach achieves high clustering accuracy and that the results obtained by combining multiple experts are generally better than those that can be obtained by individual experts. Another goal is to empirically show that CONFHEX is an improvement over CHEX.

We apply our approach to both the page clustering task and the data clustering task, two tasks which we described in the previous chapter. For the page clustering experiments, we manually label all pages spidered from web sites. Each page is assigned a label representing its page-type. We then evaluate the page clusters against these labels. For the data clustering experiments, we use a set of agents previously built using a supervised wrapper induction tool. Each of these agents extract product information from a particular site. We evaluate the data clusters against data extracted by the agents.

6.2 CONFHEX on the Web Dataset

To evaluate our approach in the data extraction domain, we run two sets of experiments. In the first set, we cluster pages into their page-types, such as detail pages or list pages. In the second set of experiments, we cluster text segments so that each cluster corresponds to a column of relational data.

6.2.1 Page Clustering

In this section, we report our results on page clustering performance. Our experiments show that the system achieves a micro-averaged pairwise-F1 score of 0.83 and that clustering accuracy is higher when all experts are active compared to when any one expert is active.

In this set of results, we use a commonly used (e.g., [14, 25, 69]) performance measure in clustering work, namely the pairwise-F1 score. The pairwise-F1 score is suitable when cluster labels of *all* samples are known. This is the case in evaluating the page clusters where we have manually labeled all pages according to their page-types, but is not the case in evaluating data clusters because we know labels only for fields extracted by the wrappers.

In computing the pairwise-F1 score, the clustering problem is evaluated as a classification problem on pairs of samples. Each pair is labeled as either "in the same cluster" (positive) or "in separate clusters" (negative). The output clustering is evaluated on how well it matches these labels on pairwise decisions. This gives counts on true positives and negatives, and false positives and negatives. From these counts, the standard F1 score is computed.

For the web domain, we experiment with a collection of pages from on-line retail stores. We collect sets of pages from these sites by both directed and random spidering, so that the sets include a variety of page-types. Then we manually label all the pages with their page-types. In the experiments, we hold out one site at a time, train our system on the remaining sites and then evaluate the system on the held-out site. We use the pairwise-F1 measure to evaluate clustering accuracy and calculate averages over the sites.

In our experiments, we compare the clustering performance of individual experts with that of all the experts combined. Our results are summarized in Table 1. The results in the "all" column are obtained by running the system with all the experts. The remaining columns contain results obtained by enabling one expert at a time. The results indicate that the clustering accuracy increases when multiple experts are combined.

6.2.2 Data Clustering

To evaluate data extraction performance, we use labeled data which we collect using manually built agents. We start with a set of existing agents that extract product information from web sites. These agents navigate through category and list pages of various retail

Site	All	Base URL	Layout	List	Table	Template	URL
antonline	61.4	1.1	62.1	1.5	36.5	18.3	**68.3**
buy	95.0	97.1	**97.9**	94.3	86.4	2.8	97.1
compusa	**100.0**	**100.0**	99.7	74.7	99.7	2.2	89.7
gateway	77.8	10.0	49.8	6.7	**87.1**	81.3	46.4
newegg	**95.6**	93.6	65.1	85.2	65.1	0.0	80.6
overstock	**88.4**	50.4	42.4	49.0	76.3	4.2	54.2
pcnation	**98.2**	78.8	59.5	0.2	77.7	21.7	92.3
photoalley	78.3	**98.4**	73.6	45.7	80.8	67.6	71.6
restockit	**54.7**	7.2	40.5	1.4	42.0	22.6	39.8
tigerdirect	83.8	**95.9**	78.8	65.6	68.0	42.0	82.1
average	**83.3**	63.3	66.9	42.4	72.0	26.3	72.2
p-value		0.074	0.018	0.002	0.018	0.001	0.027

Table 6.1: Clustering Web Pages — The goal of this experiment was to show that combining multiple experts leads to better results overall than using individual experts. The scores are pairwise-F1 scores as percentages. Bold font indicates the best score of each row. In the last row, we report p-values obtained via the paired two-tail Student's t-test. The p-value is the probability of observing the reported difference in the F1-scores with the assumption that there is no change in the performance of the system. Thus, lower p-values indicate that the results are unlikely to be coincidental.

sites and extract product information, such as name, description, price, availability and specifications from detail pages.

One issue with using agents is the mismatch between units of extraction. In general, agents extract data that does not necessarily lie on text segment boundaries, but the clustering approach does. This is because the clustering approach takes text segments as samples. As a demonstration of the problem, take the following HTML code:

```
<td>Dimensions: 10" x 2" x 12"</td>
```

From this HTML code, an agent has the option of extracting the dimensions as three separate fields. In contrast, the data clustering approach works on text nodes from the HTML parse tree and the string 'Dimensions: 10" x 2" x 12"' is an indivisible sample.

To work around this issue, we generate labels for text segments from extracted data as follows: For each text segment, we find all fields that are extracted from it, even if only part of the extracted data is contained within the text segment. Next, we compose a label for the text segment as a sequence of field names. Thus, the text segment in the example above

would be assigned a label of "Length, Height, Width" assuming that the fields extracted by the agent are "Length", "Height", and "Width". In practice, most extracted data fields correspond uniquely to text segments. Thus, most of the labels are actually sequences of length one.

The labeling process assigns a label to only some of the text segments because agents extract only information relevant to some given extraction task, e.g., these agents do not extract product reviews even when they are available on the site.

We evaluate the clustering results as we do in Chapter 4. In particular, for each agent we partition all labeled text nodes into target clusters such that a pair of text nodes are in the same cluster if and only if they have the same label. Then we compute precision, recall and F1 scores on each target cluster. As before, to compute these scores, we find the output cluster that contains the most number of target values and break ties by picking the smallest cluster. Note that even though each such decision is biased towards achieving the highest recall score, the overall evaluation is still "fair" because the clusters in the output are disjoint. In other words, it is not possible to trivially achieve perfect scores by creating a clustering of all possible clusters. After selecting an output cluster for the given target cluster, we compute precision, recall and F1 scores by taking the target cluster to be the relevant set and the output cluster to be the retrieved set, and applying the standard formulas.

Figure 6.1 shows the distribution of precision, recall and F1 scores for all target clusters. CONFHEX performs well on approximately 70% of target clusters where it achieves F1 scores over 0.9. On the remaining clusters, F1 scores vary from 0.2 to 0.9. Next we analyze the output of CONFHEX in more detail both on samples it performs well and others where the scores are lower.

In the listing below are some sample clusters and values for an agent where CONFHEX performs well. Samples are listed one per line and are grouped by the clusters that CONFHEX outputs. Each line contains the text of the sample, followed by the name of the HTML page from which it was extracted and the label that was assigned using the agent. In clusters 1 and 2, CONFHEX's output is in agreement with the labels assigned by the agent. The values in clusters 3, 4 and 5 have not been assigned any labels because these values are not in any field that this agent extracts. Cluster 4 is interesting in that CONFHEX has grouped values from different date fields, year, month, and day, into a single cluster. Analysis of these pages show that these values are from adjacent multiple drop-down menus which the DOM Template expert interprets to be a list of menus each

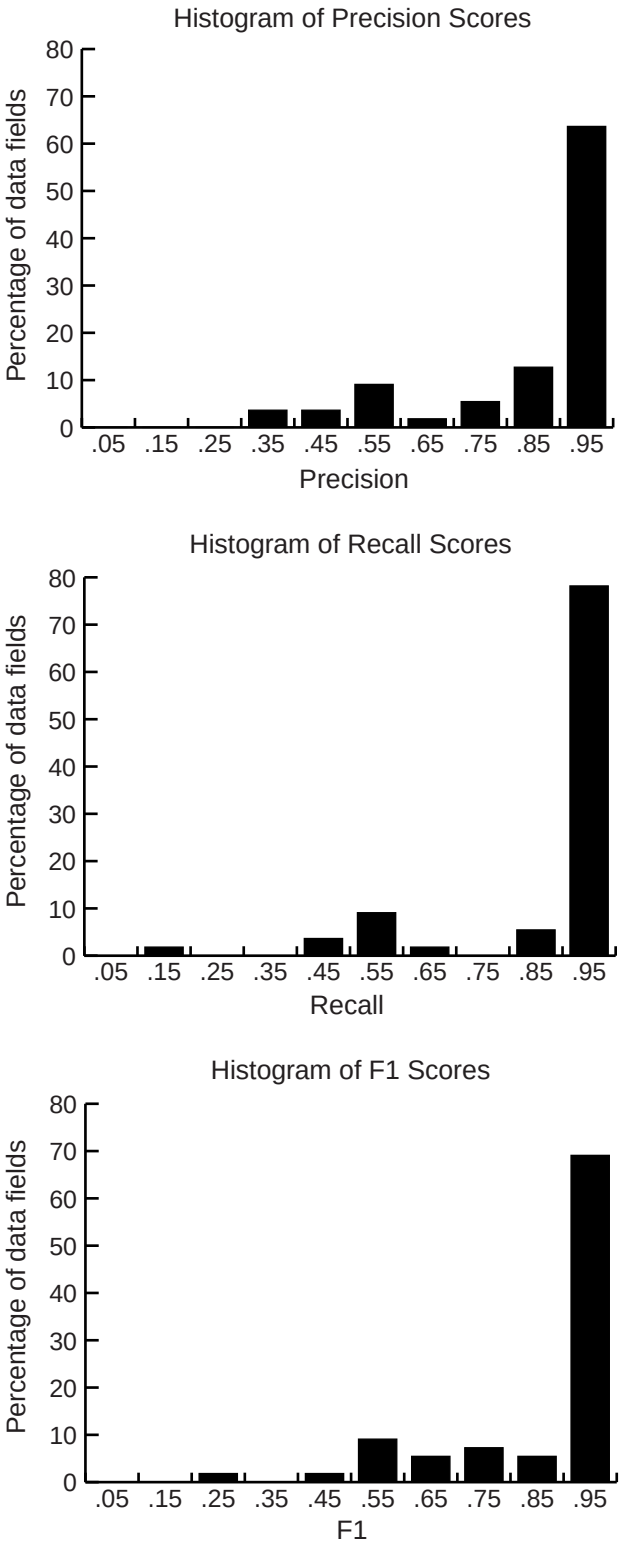


Figure 6.1: Data Clustering Results — Histogram of precision, recall and F1 scores in bins of width 0.1.

of which is a list of values. With this interpretation, the DOM Template expert generates strong hints to place these values in the same cluster. Values in cluster 5 are values from a drop-down menu. CONFHEX correctly separates this cluster from cluster 4. Cluster 3 is essentially noise where separator characters have been clustered together.

```
cluster 1:
  32.00        [Page15.html] [Price]
  35.00        [Page3.html] [Price]
  30.00        [Page1.html] [Price]
  26.00        [Page12.html] [Price]
  35.00        [Page5.html] [Price]
  ...

cluster 2:
  Traditional arrangement of seasonal fresh flowers     [Page11.html] [Title]
  Posy bowl with dark seasonal fresh flowers            [Page8.html] [Title]
  Designer arrangement of vibrant fresh flowers         [Page6.html] [Title]
  Elegant bouquet of seasonal fresh flowers             [Page13.html] [Title]
  The Gift Wrapping Service                             [Page10.html] [Title]
  ...

cluster 3:
  |     [Page14.html] []
  :     [Page13.html] []

cluster 4:
  July          [Page12.html] []
  1             [Page6.html] []
  19            [Page9.html] []
  July          [Page7.html] []
  2005          [Page4.html] []
  31            [Page5.html] []
  May           [Page5.html] []
  May           [Page4.html] []
  December      [Page7.html] []
  ...

cluster 5:
  Birthday              [Page1.html] []
  YellowPetal News      [Page1.html] []
  Get Well              [Page4.html] []
  Wedding               [Page6.html] []
  Other                 [Page14.html] []
  ...
```

To understand ways in which CONFHEX fails to find correct clusters, we sampled agents where CONFHEX's scores are low and analyzed the output in detail. The main reason for getting low precision scores is that CONFHEX interprets *name-value lists* such as "Price: $9.99, Tax: $0.80" as a single list (i.e., ["Price: $9.99", "Tax: $0.80"]) whereas the more usual interpretation is to separate the values into fields identified by the "names" in the list (i.e., [$9.99, ...], [$0.80, ...]).

Figure 6.2 shows a web page in which fields are displayed in a name-value list. In this particular case, both the DOM Template expert and Layout expert identify the name-value lists as lists of values from a single field whereas the correct clustering separates the price, manufacturer, model number and description fields. This of course leads to low precision

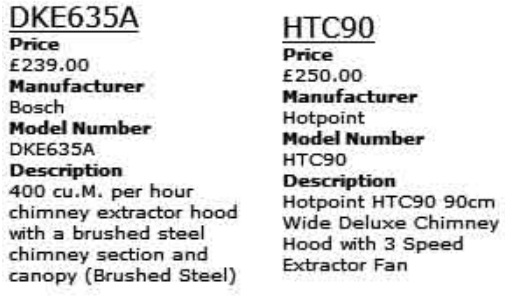


Figure 6.2: Snippet from Sample Web Page — A name-value list such as the one on this page can be represented as multiple rows in a name-value table or as a single row where the names correspond to the columns of the table.

(≤ 0.25) scores on all four fields. Below is a listing that shows a sample of CONFHEX's output for this page.

```
...
Price        [Page17.html] []
Manufacturer [Page17.html] []
Bosch        [Page17.html] [Manufacturer]
Model Number [Page17.html] []
239.00       [Page17.html] [Price]
...
```

Another reason for getting low scores, in particular low recall scores, is over-specialization of clusters. For example, when some product names are in all capital letters and some are not, CONFHEX may create two clusters for product names: one where product names are in all capital letters and one where they are not. This lowers recall scores as the evaluation picks only one of these over-specific clusters. In practice, low recall scores are less of a concern that low precision scores, as it is easier to merge over-specific clusters in a post-processing step.

Overall, CONFHEX performs well in the retail domain even though the experts we have used are generic experts with no domain-specific knowledge. In future work, we believe that we can obtain even better performance by building experts that make use of domain-specific knowledge. For example, an expert that can discover name-value pairs is relatively easy to develop in a specific domain as the "names" and "values" in a particular domain are more predictable.

6.3 Comparing CONFHEX to AgentBuilder

An interesting measure of success is the reduction in labeling effort obtained by using CONFHEX instead of a supervised wrapper induction system such as AgentBuilder. Intuitively, we except the user to do less work with CONFHEX than with AgentBuilder to extract the same data.

In measuring this reduction, a completely objective measure is difficult to obtain without user studies because the user experience in the two tasks are not identical and so simple measures such as time taken or number of mouse clicks are not representative of the actual effort. The induction system of AgentBuilder requires that the user supply one or more sample values for each field to be extracted. To provide sample data, a user needs to identify, either with mouse selection or typing, sample text for each field to be extracted. In contrast, CONFHEX generates clusters that need to be matched to fields to be extracted. Thus, labeling in CONFHEX requires the user to select a cluster of values among many clusters for each field. To compare these different tasks, we define two approximate measures of effort and evaluate the reduction in effort in terms of these approximate measures.

In the first measure, we simply count the number of labeling actions required per field on average. We analyzed the markup of 17 agents that we have used in the above experiments. On average each field has 6.6 samples. Thus, AgentBuilder requires 6.6 labeling actions per field, whereas assuming CONFHEX generates correct clusters, CONFHEX requires only 1, namely the action of picking the correct cluster for the given field. The reduction in labeling effort is therefore approximately 6.6:1.

The first measure assigns the same weight, 1.0, to the action of highlighting sample text on an HTML page and the action of picking a cluster among many clusters. Intuitively, the weight of the action should be proportional to the difficulty of performing that action. We approximate the *difficulty* of an action by the number of bits that would be required to encode the result of that action. Thus, we define the difficulty of an action as $\log_2 c$ where c is the number of choices from which a selection has to be made. The total labeling effort is then $sum_a \log_2 c_a$ where a varies over the required actions and c_a is the number of choices for that action.

In AgentBuilder, we take the number of choices to be the number of text nodes on the page being marked-up. For example, if a field requires two training samples from two pages and one page has 128 text nodes and the other has 256 nodes, then the total labeling effort for this field is $log_2 128 + log_2 256 = 15$. Similarly, in CONFHEX, we take

the number of choices to be the total number of clusters that are in the optimal clustering found by CONFHEX.

We analyzed the total labeling effort required for training AgentBuilder and selecting clusters in CONFHEX for the same set of agents as before. To do this, we evaluated total labeling effort for all the fields and then computed the average labeling effort per field. In AgentBuilder, the average labeling effort per field is 44.5 bits. In CONFHEX, the effort per field is 6.7 bits. The ratio of these is 6.7 which is just a little over the ratio we obtained by the first measure. That these ratios are close is surprising at first but is explained by the fact that the first measure does not take into page size or number of clusters. Thus, it varies widely compared to the second measure and coincidentally happens to be in the same range as the second measure for this set of agents.

6.4 Comparing CONFHEX to RoadRunner

We also compared CONFHEX to RoadRunner, an unsupervised extraction system. RoadRunner induces a grammar from a set of sample pages and then uses the induced grammar to extract data from pages that have been generated from the same grammar. To compare CONFHEX to RoadRunner, we measured the total labeling effort that would be required to build wrappers using RoadRunner. Even though RoadRunner is an "unsupervised" approach, in practice, there are two steps during which RoadRunner requires supervision. The first step is in clustering pages according to page-type. This is needed because RoadRunner requires its input pages to have been generated from the same grammar, thus the sample pages need to be grouped into page-types before being fed to RoadRunner.[1] The second step where RoadRunner requires supervision is in filtering the output. Just like CONFHEX, RoadRunner extracts all data from the set of pages in its input. However, only some of those fields contain data that is useful to the user. Thus, the user has to pick the fields that are of interest among all the extracted fields. To compare RoadRunner to CONFHEX and AgentBuilder, we measured the labeling effort for the two steps and compared it to our measurements of the labeling effort required in CONFHEX.

For our experiments with RoadRunner, we started with the 17 agents that we have used above. These 17 agents contained a total of 75 different wrappers, each of which corre-

[1]When pages from multiple page types are given to RoadRunner, RoadRunner will still induce a grammar. However, the data extracted with this grammar is not generally useful, as the grammar cannot capture the similarities between the pages. In the extreme, the whole page becomes a single data field.

sponds to a particular page-type. Among these 75 wrappers, only 59 had more than one sample page. RoadRunner cannot induce a grammar when the input set contains only one sample page, because the heuristics in RoadRunner's grammar induction algorithms rely heavily on common substrings across multiple pages. Thus, we evaluated RoadRunner only on 59 page-types.[2] When we ran RoadRunner on these 59 page-types, RoadRunner appeared to hang on 22, so we terminated the test after 2 minutes of execution on a 2.4 GHz processor. On the remaining 37 page-types, RoadRunner was able to induce a grammar within a few seconds. We then counted the number of fields, which we use in computing the labeling effort, that RoadRunner extracted using the induced grammar.

To compute the labeling effort in using RoadRunner, we used the following process:

- Collect all necessary pages from the site. We assume grammar induction will need as many sample pages as have been used in AgentBuilder to induce extraction rules. For each wrapper, we find this number from the hand-built agent and pass as input to RoadRunner the same sample pages that have been used to induce extraction rules.
- Classify each page into its correct page-type. Each such decision requires choosing one page-type among n page-types. The number of page-types is determined from the number of wrappers in the hand-built agent. This is the first manual decision making step.
- Run RoadRunner on each page-type.
- Among all the fields RoadRunner extracts, pick the columns that are of interest. For each field, this requires picking one from n-fields where we compute n by examining the output of RoadRunner. This is the second manual decision making step.

From the data we collected by running RoadRunner on the 37 page-types, we computed the labeling effort to be 8.6 bits per field. Of that amount, 2.6 bits come from clustering pages into the correct page-types, and 6.0 bits come from selecting fields of interest among all extracted fields. These numbers (Figure 6.3) match our expectations: Using RoadRunner requires less effort (8.6 bits/field) than using AgentBuilder (44.5 bits/field), but more than using CONFHEX (6.7 bits/field).

A second observation is that the extra effort in using RoadRunner comes from the fact the input pages need to be sorted into correct clusters before being fed to RoadRunner. Even though this suggests that an alternative to CONFHEX is running a page clustering

[2]This shortcoming of RoadRunner can be addressed by collecting sample pages at different times, assuming the data on the page changes over time.

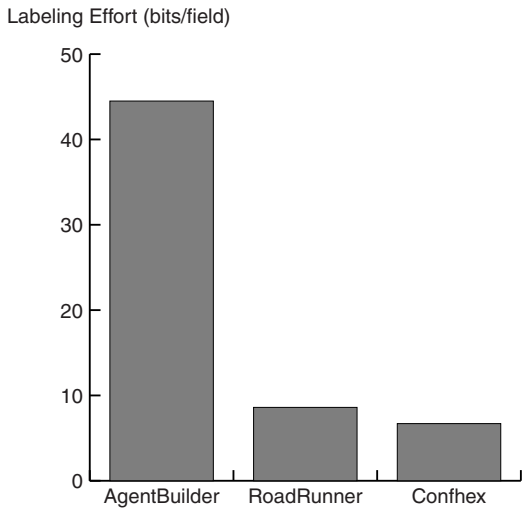


Figure 6.3: Levels of Effort — We compare the levels of effort required to extract data from the web using three different approaches: AgentBuilder (a supervised rule induction system), RoadRunner (an unsupervised wrapper induction system), and CONFHEX (an unsupervised site-extraction system)

algorithm before running grammar induction on particular page-types, the clustering accuracy of a human would be difficult to replicate without actually "understanding" the contents of the pages, which is essentially what CONFHEX does.

6.5 Comparing CONFHEX to CHEX

We developed CONFHEX, motivated by our observations of the shortcomings of CHEX. In this section, we experimentally test whether CONFHEX performs better than CHEX.

One significant difference between CHEX and CONFHEX is that in CONFHEX hypotheses have confidence scores whereas in CHEX they are boolean-valued. Ideally, we would like to compare CHEX and CONFHEX by evaluating them on the exact same dataset. Unfortunately, the two systems work on different units of text (CHEX on token sequences and CONFHEX on DOM text nodes) and generate output in different forms. This means that the effort required to accurately compare the two systems is prohibitive. Instead, to verify that confidence scores lead to an improvement in performance we decided to simulate CHEX using CONFHEX. To do this, we modified CONFHEX so that the confidence scores generated by CONFHEX are rounded to either 0 or 1 depending on whether they are less than or greater than the neutral confidence score of 0.5. We call the modified system

CONFHEX0-1. We then ran both CONFHEX0-1 and CONFHEX on a set of agents. Table 6.2 shows the precision and recall scores obtained by the two systems on a set of agents. In general, CONFHEX appears to achieve better precision scores at about the same recall level. Statistical analysis confirms that with CONFHEX, precision is better than that with CONFHEX0-1 at 5% significance level using the Wilcoxon signed-rank test (z=2.09)[3] and differences in recall are statistically insignificant.

6.6 Applications

In this section, we demonstrate the use of CONFHEX in a particular application domain, namely news extraction, where the task is to extract the date, author and text of news articles from news sites.

6.6.1 News Extraction

The goal in the news extraction domain is to extract news articles from a news or blog site. More specifically, given a page that contains, among others, links to news stories, the goal is to extract the title, the body, the author and the date of each news story. In this section we demonstrate the use of CONFHEX to tackle the news extraction task.

For this set of experiments, we built agents using AgentBuilder to extract headlines, article bodies and dates from four different newspaper web sites. We use two of the agents to train the system and the other two to test. Then we switch the two sets of agents for training and testing, so that we get data for all four news sites.

Below is some sample data from clusters that CONFHEX has found on Pittsburgh Post-Gazette's web site. Headlines are correctly clustered (cluster 2). Dates have good recall but date-like strings appearing in some text nodes are wrongly clustered into the date cluster (cluster 4). Providing more training data is likely to improve the accuracy of the confidence scores for the date expert and reduce this type of error. Paragraphs of article bodies are also correctly clustered (cluster 5). However, this cluster requires some post-processing to more accurately represent data, because each article body has been split into multiple paragraphs and ideally these paragraphs need to be grouped together into article bodies. The remaining clusters demonstrate other clusters that are not relevant to this extraction task but have been identified by CONFHEX.

[3]The test assumes the distributions have the same shape and spread

```
cluster 1:
  Rates          [Page8.html]  []
  Vacation       [Page3.html]  []
  Rates          [Page10.html] []
  What's New     [Page2.html]  []
  ...

cluster 2:
  How they voted                                                  [Page1.html] [Headline]
  Senate OKs bill to update mine safety measures                  [Page1.html] [Headline]
  Hearing held to decide on new trial in 1989 slaying             [Page1.html] [Headline]
  Jefferson Hills man gets 20 yrs. for distributing drugs          [Page1.html] [Headline]
  ...

cluster 3:
  $530                  [Page7.html] []
  PG.GoogleAd();        [Page2.html] []
  OAS_AD("x03");        [Page3.html] []
  $174,900              [Page10.html] []
  ...

cluster 4:
  (02/12/2008)                                  [Page1.html] []
  Sunday, February 10, 2008                     [Page10.html] [Date]
  Monday, February 11, 2008                     [Page9.html] [Date]
  In addition, ... 2008-2009 ... years.         [Page8.html] [Body:]
  ...

cluster 5:
  There was never a groundbreaking date ...     [Page5.html] [Body]
  "All of the plays are darkly comic ...        [Page2.html] [Body]
  "Click" concerns a husband who entertains ... [Page2.html] [Body]
  ...
```

These initial experiments showed that the application of CONFHEX in the news extraction domain can be improved in two areas. First, by adding domain specific experts and providing more training data, we believe CONFHEX can achieve higher accuracy in the news extraction task. Second, by trading off the generality of CONFHEX for extraction performance, the system can be made to run faster. In the rest of this section, we discuss the trade-off between generality and performance in more detail.

One alternative to directly applying CONFHEX is to run CONFHEX off-line to analyze and understand the structure of the site and then build an agent using a supervised wrapper induction approach but providing the automatically discovered structure as training data. The resulting agent can then be executed for high-performance extraction. In general, this is a useful approach to automatically build agents, which maintains build-time generality and trades off run-time generality for performance.

Another alternative is to take advantage of the fact that a large percentage of news sites have the same simple structure: each site has a number of "sections", such as "world", "U.S.", or "technology", each of which contain a list of links to news articles. Starting from these section pages, the news extraction task is to identify links on section pages that

point to news articles and then to identify the author, date and text of articles on the article pages.

The news extraction task, when defined much more narrowly compared to the site extraction problem, still benefits from sophisticated, heterogeneous experts but allows simpler methods in combining the experts. For example, for the article URL identification sub-task, experts can analyze each URL within the context of list pages, using information obtained from complex structures such as layout and DOM structure, much like the experts of CONFHEX. Combining the output of these experts to determine which URLs link to news articles is then a task of classification, rather than clustering, where the input to the classification step is the output of the experts. Extracting fields from the article pages can be done in a similar fashion: experts analyze the article page so that finding the value of a field becomes a classification task given the output of all the experts. Unlike CONFHEX and the first alternative discussed above, this approach only works on particular types of news sites, but has the advantage of being simpler overall.

The news extraction application is interesting not only in that we are able to use CONFHEX directly to perform automatic extraction, but also in that we can trade off the generality of CONFHEX for performance to satisfy real-world constraints.

Site	Precision CONFHEX0-1	Precision CONFHEX	Recall CONFHEX0-1	Recall CONFHEX
Appliance4U	25.60	25.40	100.00	99.00
ApplianceDirect	25.60	30.90	74.20	73.20
AuchanHistoryPoliticsBooks	53.20	100.00	100.00	100.00
BumpsMaternity	49.80	56.60	97.20	95.10
BuyDiscountPerfumes	17.30	15.60	88.60	88.10
Castorama5	30.00	33.20	71.80	94.00
CcSportsAndFitness	83.30	55.30	82.50	73.10
CdExpressL	20.20	20.30	99.00	99.00
CdExpressS	16.90	12.60	97.40	97.40
CdExpressWXYZ	16.90	16.90	98.60	97.40
DecathlonFr	72.60	56.20	81.20	69.40
DigiUk	100.00	100.00	65.20	100.00
EpFashions	50.00	100.00	94.20	100.00
Etam	14.40	14.40	100.00	100.00
FamoustoreDVDs	100.00	100.00	55.80	44.80
FergusonFitness	31.80	20.00	100.00	89.30
Flowers800	26.60	47.50	95.40	96.60
FrontRowDvd	20.80	20.70	93.00	99.70
GardenBuildingsDirect	53.80	53.90	96.80	95.80
GeraldOnline	38.00	47.60	100.00	96.60
GreatSoftwareOnline	81.30	25.50	98.60	99.30
HiEnergyShop	36.40	27.10	99.40	99.40
HqHair	53.60	57.60	97.90	63.50
InternetCamerasDirectAccessories	53.60	56.60	92.60	95.90
Joueclub	15.70	26.60	96.30	98.40
LookFantastic	23.00	38.40	78.20	75.80
MattressOnline	53.40	53.40	100.00	100.00
MillerPC	33.30	93.10	57.40	52.60
Mx2	49.70	86.30	83.20	50.70
Natoora	24.40	60.10	78.50	81.00
PixelFlash	21.40	37.30	92.00	86.40
Quelle	20.00	20.00	90.50	92.30
SaverSoftware	86.80	86.80	100.00	100.00
Scarlett4U	40.80	54.00	92.90	88.20
ShoeShop	48.20	61.80	93.40	96.80
SnapdragonJewellery	70.90	44.80	100.00	100.00
SportE	44.00	46.20	72.70	75.20
TackleShop	100.00	100.00	90.60	100.00
TrueShopping	58.50	40.40	99.30	77.40
TvAndVideoDirect	39.90	54.10	96.80	100.00
UKAppliances	21.30	38.70	54.60	49.80
WaitRoseDirect	60.40	79.40	88.50	83.00
WowWoman	17.70	22.90	90.10	93.90
YellowPetal	66.70	58.30	85.70	100.00
unemilleordifr	25.60	55.50	93.20	95.20

Table 6.2: Comparing CHEX and CONFHEX — We simulated CHEX using a modified version of CONFHEX to compare the two systems. CONFHEX achieves better precision scores than CHEX.

Chapter 7

Record Deduplication

7.1 Overview

In the previous chapters, we developed a framework for clustering web data with heterogeneous experts and used this approach to automatically extract data from web sites. We evaluated our approach using data collected by supervised wrappers, but we were not able to do a comparative analysis because of lack of standard site-extraction datasets. In this chapter, we apply our approach to record deduplication, where we compare our results to some published results.

Record deduplication is the problem of identifying records that refer to the same entity. The challenge in record deduplication is to be able to do this even when records contain noise such as misspelled, missing, incorrect or ambiguous values. For example, does the record "Name: John Smith Address: 123 Main St" refer to the same person as "Name: Smith, John Address: 132 Maing Street"? Making a good guess to answer this question involves understanding that "John Smith" and "Smith, John" are the same names, that "John Smith" is a common enough name that it does not necessarily uniquely identify a person, that "St" is an abbreviation for "Street", that depending on whether 123 and 132 are legitimate street numbers on "Main Street", the digits 2 and 3 might have been unintentionally transposed, that "Maing" is a misspelling of "Main" and so on.

Even though the name record deduplication implies finding and removing records that are identical in a given dataset, the techniques used in record deduplication are also useful in aggregating data from multiple data sources. In aggregating data, it is also useful to identify records that refer to the same entity so that the relations between the records

can be correctly established. When cast as the problem of linking records from multiple data sources, record deduplication is also referred to as record linkage. The record linkage problem has gained importance as many new data sources have become accessible through the web. For example, if one data source contains product reviews and a second source contains pricing information, it is useful to be able to link the two data sources together so that there is a unified set of products and to each product its correct review and price are assigned. If each product is assigned a unique identifier and both data sources include the identifiers, then clearly the problem is trivial. Unfortunately, this is rarely the case. Either there are no unique identifiers or the identifiers are not valid across multiple data sources. Thus, the problem is identical to record deduplication.

In our experiments in the record deduplication domain, we analyze citations, which consist of individual fields such as authors, title, and venue, each of which has their own characteristics. We identify duplicate citations by clustering the records such that each cluster represents the set of duplicate citations. In doing this, we use experts that understand how the value of a particular field from duplicate citations may be represented in different forms, e.g., abbreviations of conference names.

There has been a great deal of research in the record linkage domain. Most of this work (e.g, [5, 61]) focuses on the pairwise record-linkage problem formalized much earlier by Fellegi and Sunter[31], where each pair of records is considered independently. In Fellegi and Sunter's formalization, a file of records A is merged with another file of records B. During the merge, every record in A is compared against every record in B. The goal of the comparison is to determine whether the record in A represents the same entity as the record in B. The comparison decision is a function of a comparison vector of features on pairs of records. For example, one component of the comparison vector might represent whether a particular field of the two records contains identical strings. Fellegi and Sunter show how to determine an optimal comparison function by minimizing errors that the resulting record linkage rule makes.

In some recent work (e.g, [25, 57]), the focus has shifted to modeling the record-linkage decisions jointly. The main difference between Fellegi and Sunter's formalization and the recent work is that whereas in the former the decision for each pair of records is made independently of any other decisions, in the latter the decision for a pair has influence over decisions on other pairs. To see why the latter approach is more powerful, consider the case where record a is compared against records b_1 and b_2. If a can be linked to b_1 and b_1 can be linked to b_2, then intuitively there is additional support for linking a to b_2.

Fellegi and Sunter's formalization does not take into account such support whereas the recent work in modeling decisions jointly does.

The application of our approach to the record linkage domain is more closely related to this type of modeling. In particular, it is useful compare our approach to Culotta and McCallum's recent work [25]. Both approaches model pairwise similarities in a probabilistic framework and use greedy agglomerative clustering to find a locally optimum solution. The main difference between the two approaches is that Culotta and McCallum focus on learning feature coefficients jointly by using conditional random fields whereas we use a simpler Bayesian model and train experts independently.

An advantage of our approach is that it is much easier to add new experts. Because each expert is trained independently, adding new experts is not computationally demanding. In contrast, adding a new expert to a joint-model involves retraining all the experts.

We used the same experimental set-up as Culotta and McCallum, based on a dataset from the Cora Computer Science Research Paper Engine. The Cora dataset is a collection of citation records, where each record consists of a number of fields such as title, author, date, journal, etc. The record linkage problem is to identify those records that refer to the same citation.

To cluster records, Culotta and McCallum define a set of features over pairs of samples. Their features are based on string-edit distance of individual field values and also of the full string representation of a record. By using simple features, Culotta and McCallum leave it up-to their learning algorithm to discover what is already known about the citation domain. Thus, their approach requires a framework that can learn complex relationships between simple features.

This kind of approach may be useful in some domains, but most real-world problems don't require such complex frameworks if more background knowledge about the problem can be made available to the solver.

Like Culotta and McCallum, we also solve the citation and venue deduplication problems, and do so without co-clustering, and compare our results to theirs. In contrast to their string-edit based features, we developed several knowledge-rich experts that capture some of the common types of structures that we observed in the citation domain. We were able to obtain good results by solving the citation and venue problems independently. Investigating whether co-clustering with knowledge-rich experts provides additional improvement remains as future work.

As our baseline, we also evaluated our approach using a generic expert that measures

string similarity between pairs of samples. We created multiple instances of this expert for individual fields. Thus, we have experts for fields such as author, title, and date. This is similar to Culotta and McCallum's approach except that the learning approach is considerably simpler.

The next two section describe the experts we have used for clustering citations.

7.1.1 Citation Experts

- **Normalized Title**: Many variations of titles are due to differences in spacing, capitalization, and truncation, so we applied string-edit distance to titles after we removed spaces, lower-cased all letters, and truncated long titles to a fixed number of characters.
- **Venue Type**: In the Cora dataset, some fields are optional. In particular, there are several different fields, e.g., journal or book-title, that contain the venue name and each record has only one of these fields present. One of our experts simply identifies whether two records have the same venue field or not. Note that this information is not available by string-edit distance on corresponding fields.
- **Venue Name**: One of the experts compares the venue name regardless of which of the several venue fields it comes from after normalizing it by removing spaces and common stop words, and lower-casing it.
- **Book Title**: The book title field is overloaded in that it is sometimes the title of the book in which the cited work appears and sometimes the title of the book which is being cited. In the former case, the record also contains a title field whereas in the latter it does not. Thus, the expert which compares the book title field uses the value in this field only when the title field is not present.
- **Journal Article**: Most citations to journal articles contain two fields, volume and page, in addition to the journal field. One of our experts uses all three fields in a hierarchical way to compare citations. If all three fields match, then similarity is high; if only journal and volume match, then similarity is lower; if only journal matches, then it is still lower; if no fields match; then it is lowest. Before comparing the individual fields, the expert also normalizes the journal name (in the same way as titles are normalized), and the volume and page fields by removing all characters except the digits.

- **Location**: This expert simply computes string-edit distance on the location field.

7.1.2 Venue experts

- **Venue Abbreviations**: This expert checks whether one venue name is an abbreviation of another name when one name is less than four characters long and the other is more. For the shorter name to be an abbreviation of the longer one, each of its characters must appear as the first character of some word in the longer.
- **Editor**: This expert simply computes string-edit distance on the editor field.
- **Normalized Title**: Same expert as above.[1]
- **Venue Name**: Same expert as above.

7.2 Experiments

Culotta and McCallum approach the citation-deduplication problem as a co-clustering problem where they simultaneously cluster the dataset to find co-citations and also cluster it to find co-venues. This is useful because identifying co-citations provides additional information for finding co-venues and vice versa. Thus, they report two sets of results: baseline results for clustering citations and venues independently and improved results from co-clustering them.

Following Culotta and McCallum's experimental design, we evaluate our system on each of the three hold-out subsets,[2] use the pairwise-F1 measure to evaluate clustering accuracy, and report micro-averages over the pairs.

7.2.1 Results

Our approach achieved pairwise-F1 scores of 0.938 and 0.891 in the two sub-problems of the dataset. These scores are higher than Culotta and McCallum's scores of 0.908 and

[1]Note that even though we use the exact same expert for both the citation and venue problems, the confidence scores assigned to a pair of samples by the same expert will be different in the two problems. For example, a matching journal name indicates co-venues with high probability but co-citations with a much lower probability.

[2]The running time of our system varied between a few minutes to half an hour on a 2.4 GHz Pentium. Culotta and McCallum report running times from 20 minutes to an hour, but experiments were done on different hardware, so performance results are not comparable.

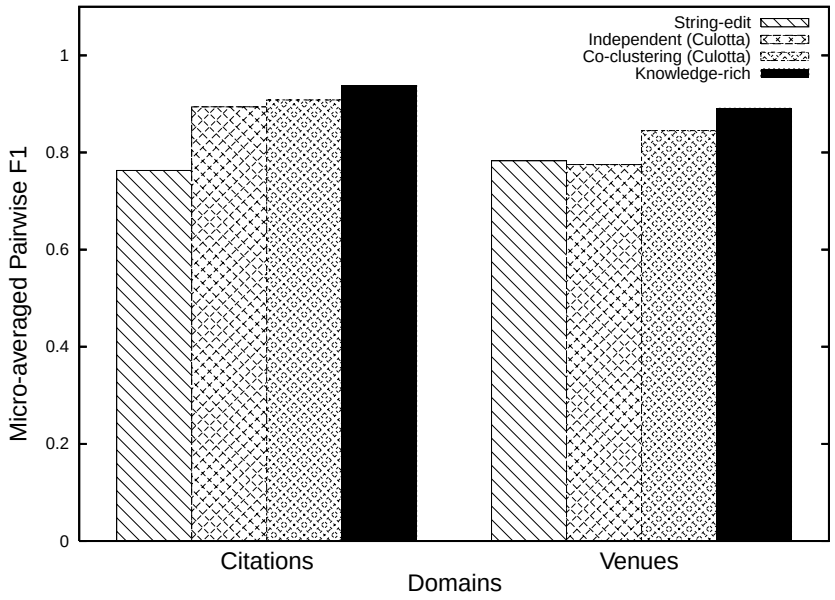


Figure 7.1: Record Linkage Results — Tacking the citation deduplication problem with knowledge-rich experts gives results comparable to those obtained with a more sophisticated and computationally-expensive framework.

0.845 on the two datasets, but additional experiments using both systems would be needed to infer that one system is better than the other. Figure 7.1 gives a summary of the four sets of results: Our baseline with the generic string-edit distance, Culotta and McCallum's independent clustering and co-clustering, and our results with knowledge-rich experts. There are two observations to make. First, our baseline is lower than Culotta's independent clustering results in the citation problem but not in the venue problem. We believe this is because the venue problem is simple enough that the more powerful representation of Culotta's approach does not provide any gains over our simpler approach. Second, using knowledge-rich experts with a naive-model gives better results than that of learning a joint-model of simple "experts", at least on this particular dataset.

7.3 Summary

In this chapter, we applied our approach in the record deduplication domain by developing a set of experts specific to this domain. Our results demonstrate that combining experts that analyze multiple types of structure compares well to a framework that learns complex interdependencies between simple features.

Chapter 8

Related Work

8.1 Data Extraction from the Web

Data extraction from the web is a well-studied area of applied artificial intelligence research. Much of the previous work focuses on finding new ways to extract data from the web pages or improve the quality of existing techniques. Our work also follows this path, but rather than focusing on a particular technique, our goal is to build a framework in which many of these techniques can be combined in a principled fashion. In this section, we describe some of the common techniques used for data extraction from the web.

8.1.1 Wrappers

A wrapper is a function that maps web pages to extracted data[48]. The wrapper induction algorithms take as training examples a set of labeled pages and typically find regular-expression like patterns to locate the labels within the pages.

Extensions to the basic wrapper, such as in Stalker[46], are possible so that the induced wrapper extracts structured data instead of labels. The input to Stalker is the *embedded catalog*, which defines the hierarchical structure of the data in terms of lists, groups and items, and a set of training examples, which are labeled pages. A page is labeled by marking up the data to be extracted on the page and linking it to its role in the embedded catalog. Stalker then learns a set of extraction rules using a covering algorithm and attaches them to the embedded catalog. The embedded catalog together with the attached rules is a wrapper that can extract structured data.

Wrapper induction systems like Stalker require no programming and are very useful

in reducing the time to generate a wrapper by hand, but they have several shortcomings: First, an induced wrapper typically works on a single type of page. Second, a wrapper makes available only the type of data that it is trained on (e.g., if a wrapper is built to extract zip codes, it will not extract state abbreviations). And most importantly, wrapper induction requires human effort to create training sets.

In contrast to wrapper induction systems, our approach can wrap a web site without any training sets and will turn all data that is on the site into machine processable form. In fact, the output from our approach can be viewed as the result of building and applying a consistent set of wrappers to a given site.

Wrapper induction can also be cast as a classification problem[19]. In this type of approach, an HTML page is first represented as a DOM tree. Each DOM node is then assigned a label which represents whether the content of the node is extracted or not. The classification problem is to assign to each node the correct label. When extraction is cast a classification problem, standard machine learning algorithms can be used to automatically build a classifier from training data. The resulting classifier can then be applied to new pages to extract data from unseen pages. This type of approach requires each DOM node to be represented as features. Cohen and Fan define 19 such features. These features represent the local structure of the DOM tree, such as the name of the parent node, the number of child nodes and so on. The advantage of this approach over the more traditional wrapper induction approaches is that the induced classifier can be made more general than a traditional wrapper in the sense that it can successfully label nodes from different page types. This is because the hand-coded features are less sensitive to page-type than the actual extraction patterns that wrapper induction algorithms learn. Our approach is similar to Cohen and Fan's approach in that the types of structure that the algorithms work on are at a higher level of abstraction than individual tokens. However, while Cohen and Fan's approach uses hand-coded features over the DOM tree, our approach raises the level of abstraction from features to arbitrary experts, allowing multiple types of representation such as page layout and link structure to be utilized.

Another dimension in which wrappers can be generalized is the page representation over which extraction patterns are learned. Cohen, Hurst and Jensen introduce a system in which the wrapper induction system is composed of a master algorithm which learns extraction rules over multiple languages each of which is defined by a *builders*[20]. Each builder defines an extraction language over a different representation, such as token sequences, DOM trees, or page layout and also a method to minimally generalize an extrac-

tion rule in that language to match training samples. For example, a builder can define an extraction language over the token sequence representation of a page, so that a rule to extract the price of a product from HTML segments such as "Price: $7.99" can simply be expressed as the tokens following "Price:". The master learner than induces extraction rules from training samples by a set-covering algorithm that combines the extraction rules from the builders. This approach and our approach shares the intuition that for successful extraction over many types of page-types and web sites, it is necessary to take advantage of different types of structure. Whereas Cohen, Hurst and Jensen focus on inducing wrappers by directly utilizing different representations of pages, we focus on discovering the relational form of the page indirectly from the hypotheses generated by analyzing different types of web structure.

In a recent survey of web data extraction approaches[12], Chang et al., compare current approaches in three areas: level of structure in the input (e.g., extracting facts from free-form news articles as opposed to extracting data from HTML), techniques used in the approach (e.g., extraction rule induction or grammar induction), level of automation (e.g., whether an approach requires collecting training samples). Additionally, they categorize current approaches into four levels of supervision: manually built, supervised, semi-supervised, and unsupervised. It is useful to analyze our approach along the first two areas (the level of automation is mainly reflects the engineering effort put into turning an approach into a polished system) and level of supervision. Our approach falls into the unsupervised category, requiring no training samples for extraction. Unlike other unsupervised approaches which rely on grammar induction and require input pages to be of the same page-type, our approach works on pages from multiple page-types. In terms of the structure of input, like other web extraction approaches, our approach expects its input to be semi-structured. In particular, our approach relies on the assumption that many web sites are generated from data that is relational to start with and that the relational structure has been partially preserved in the observable structure of the site. In terms of extraction techniques, our approach is novel in that it uses a clustering approach whereas existing approaches either use induction or classification for extraction.

8.1.2 Table Extraction

Table extraction research focuses on *detection* and *understanding* of tables in text and web documents[81]. Detection involves locating tables on a page whereas understand-

ing involves segmenting tabular data into cells and aligning them correctly. The current techniques either rely on the layout of the document or on the syntax and semantics of its content. For example, it is possible to take advantage of HTML markup commonly used to display tables to find values inside a table. Furthermore, because almost all tables are formatted such that records are laid out horizontally on the page, it is valid to assume that records are composed of consecutive values as they appear in the HTML source. With these assumptions, table understanding can be cast and solved as a constraint-satisfaction problem[49]. As another example, semantic information represented in the form of an ontology can be used to perform table understanding[29, 77]. The ontology defines the group of attributes that records in a table are likely to have, the patterns that match values of attributes, and keywords that are likely to be around particular attributes. The knowledge stored in the ontology then makes it possible to identify records and their attributes on HTML pages. Alternatively, tables can be extracted by using heuristics that rely only on visual, in other words page-layout, information[33]. In general, these types of approaches[13, 47, 59, 72] focus on a particular structure and are complimentary to ours in that a table extractor can easily be turned into an expert.

In this work, we are attacking a bigger problem, which includes table extraction as a sub-problem. In fact, a table extraction algorithm can be used as an expert. Even though the problem is bigger, our approach is potentially at an advantage, because it starts with richer input that can include multiple samples of a single table structure as well as other types of structure such as URL patterns that help in extracting tabular data.

8.1.3 Grammar Induction

Starting from a set of positive and negative examples, grammar induction aims to find a formal grammar or automata for a language that contains the positive examples but not the negative ones. In general, grammar induction is a hard problem.

The site-extraction problem can be cast as a grammar induction problem where the relational structure is imposed on the space of grammars. For example, the grammar rule for a page is in the form of $page_i \rightarrow cell_1 cell_2 \ldots cell_n$ and each $cell$ rule is either $cell_i \rightarrow$ **literal** or $cell_i \rightarrow relation_j$. In this formulation, the problem is to choose the correct rules and instantiate the literals to correct token sequences.

One early grammar induction system is Sequitur. Sequitur relies on two hard constraints to limit its search space: Pairs of symbols are unique within the right side of

grammar rules and every rule is used more than once.

Another research project that is based on grammar induction and that aims to build wrappers for web pages automatically is RoadRunner[24]. In general, grammar induction is an intractable problem, but RoadRunner restricts the form of candidate grammars to turn its search space into a tractable one and can induce grammars from a set of positive examples.

Zhai and Liu[82] describe a method to automatically build wrappers for list pages by analyzing a single sample page. Their method relies on the following two assumptions. 1. Each record is contained within a unique subtree of the parse tree. 2. Collections of records are laid out in contiguous areas on the screen. With these assumptions, their method works as follows: 1. Identify collections of data records by finding subtrees that are close to each other on the screen and similar in structure. 2. Within each collection, identify records and their fields by aligning subtrees with a heuristic tree alignment algorithm. Interestingly, even though their method relies heavily on DOM structure, it does not directly use the DOM tree obtained from the HTML source of a page. Rather, it constructs a new DOM tree by combining information from the layout and parse of a page. The authors observe that a page that is displayed correctly does not necessarily have a correct DOM structure. This observation is in line with our hypothesis that discovering web page structure requires multiple types of experts.

In general, the disadvantage of grammar induction is that utilizing different types of web structure becomes difficult within a grammar induction framework. For example, to use the layout of a page during grammar induction, either the layout information has to be encoded into the samples, which normally contain only text of the pages, or the induction algorithm has to be modified in an ad-hoc fashion to make use of the layout information, perhaps as a heuristic, while it is processing the pages. Clearly, neither of these approaches is satisfactory.

8.1.4 Unsupervised Extraction

When the extraction task is restricted to a specific domain, it is possible to extract data without explicitly building wrappers. In their recent work[60, 80], both Wong et al. and Probst et al. focus on the task of extracting product attributes from web pages. The intuition in Wong's approach is that by combining site-independent, but domain specific knowledge, about products and site-dependent page-layout information, product attributes

can be extracted in an unsupervised fashion. When some of the values of attributes are known, these values can be used to understand the site-specific page-layout of a new site. For example, knowing that the attribute "resolution" takes in values such as "600 dpi" allows the discovery of specific formatting (e.g.,"italic font on the second column of the parent table") on a particular site. The discovery of the site-specific format then allows the rest of the attributes and their values to be located. Wong et al. use this intuition to develop a framework which assigns probabilities to text segments and their attributes by combining site-independent domain knowledge and site-specific layout information which are represented in a probabilistic graphical model. Similarly, Probst et al., describe an approach where attribute values are extracted via classification. To reduce the number of training samples, they use a heuristic approach to automatically label samples with high precision (but low recall) and then use naive Bayes to improve the recall. In comparison to our approach which defines a general framework for combining an arbitrary number of structures, both approaches focus on a particular extraction task and use a small number of specific types of structure: values of attributes are generally common across multiple sites and that different attributes are formatted similarly or appear adjacent to similar words on any given site. The trade-off over our approach is that the specific extraction task can be accomplished without the need to build many experts.

8.2 Relational Model Learning.

The relational learning problem is to find a model that can predict the values in a relation. The model, which can be decision trees, first order logic formulas, Markov models, etc., is built based on a given set of tuples from the relation. Once the model is learned, missing attributes can be predicted based on the values of known attributes. We will look at Rapier and probabilistic relational models (PRMs), two approaches that have been applied to the web domain, in detail.

Rapier's[10] learning algorithm is based on ideas from inductive logic programming (ILP) research. The extraction pattern language it uses is analogous to the first order logic formulas of ILP in that the patterns are generalizations of the training examples. Rapier uses a specific-to-general search to find patterns and guides its search using a compression-based metric.

PRMs[34] extend Bayesian networks from modeling flat data sets to modeling richer

relational data sets. Like Bayesian networks, PRMs are probabilistic networks that represent the statistical dependencies of attributes within a single table, but in addition to Bayesian networks, the dependencies in PRMs also include attributes from related tables. Once the parameters of a PRM are determined, it assigns a probability to any instantiation of the relational structure.

In general, relational model learning approaches are difficult to apply to the web wrapping problem, because these approaches assume that their input is from a relational source. In the web wrapping problem, the bulk of observable data is in the form of token sequences. To apply relational model learning approaches, the web sites need to be converted into relational form first.

One way to do the conversion is to use a meta-model where the relations do not model the relations between the data but between the objects, such as pages and tokens, that represent the data[35]. To capture the sequential relation between tokens, the meta-model has to introduce either some "precedes" relation or indices to label the tokens. In the first case, long range structures, such as between the header and footer of a page, are hard to discover. In the second case, the indices need to be treated specially, because withing the relational model, indices do not carry their usual ordering and closeness meaning.

A second way is to start with a known relational model for the data[71]. This approach is useful in classifying and clustering pages when the underlying relational model is known, but applying it to new sites requires additional manual modeling work.

In contrast to relational model learning approaches, where the algorithms look for a model that best fits the relational data, our approach searches for a relational representation of the (sequential and linked) data that best represents the multiple types of substructure of the data.

8.2.1 Data Mining in Graphs.

Another area of research, which is related to grammar induction and relational model learning, is data mining in graphs. Subdue[22] is a system that discovers substructures in structural data represented as graphs. Objects in the system are represented by nodes or small subgraphs, and relations between them by edges. Substructures are subgraphs that occur multiple times in the graph. Subdue builds a dictionary of substructures and replaces the occurrences, which may match only approximately to the substructures, by references to the entries of the dictionary. The process is repeated on the resulting graph

and substructures are allowed to contain references to existing substructures. In this way, nested substructures can be discovered.

The hierarchical template expert and the Subdue approach are similar in that starting from instances of a concept, both induce the concept and replace the instances with references to the concept and so are able to discover complex structures. In the case of the hierarchical template, the instances are the rows of a list and the concept is the row template. In Subdue, the instances are the subgraphs and the concept is a pattern that matches the subgraphs.

The hierarchical template algorithm is a specialized version of Subdue, because it works on a particular graph, the DOM tree, and looks for a particular kind of substructure. As in the relation learning case, applying graph mining techniques to the web wrapping problem suffers from the fact that the token sequences are not easily put into structured form.

8.3 Clustering

Clustering is the problem of partitioning data so that similar samples are grouped together. Our approach uses clustering as a framework in which multiple heterogeneous experts combined. In particular, the common language that experts use is defined over clusters. Experts output hypotheses that state how likely two samples are in the same cluster. We also use standard clustering algorithms[27], such as the leader-follower algorithm in CHEX and the greedy agglomerative algorithm in CONFHEX.

Even though our approach relies on clustering concepts and techniques, the emphasis of our approach is orthogonal to that of core clustering research. Our focus is combining heterogeneous experts in a principled way whereas the focus of clustering research is clustering algorithms[41]. Core clustering research focuses on improving clustering algorithms in performance and accuracy.

There are some specific research areas within the clustering domain which are more relevant to our approach than core clustering research. Next we discuss several relevant areas: coclustering, clustering with constraints and ensemble clustering.

8.3.1 Coclustering

In coclustering, two related clustering problems are solved simultaneously to take advantage of the fact that a cluster of samples in the first clustering problem provides some information about the clustering of related samples in the second problem. For example, when documents and words are clustered into topics, documents in the same cluster are more likely to contain words that are also grouped together. As another example, in the citation deduplication problem of Chapter 7, when two records are identified to be in the same venue, then they are more likely to refer to the same citation. In the web domain, data items are clearly more likely to be in the same cluster when the pages which the data items are on are in the same cluster.

One approach to coclustering is to model the particular coclustering constraints explicitly. For example, in the citation deduplication problem of Chapter 7, Culotta and McCallum[25] define a probabilistic model that explicitly defines the relation between two records having the same venue and two records referring to the same citation. Alternatively, a more general approach to coclustering can be taken by considering the joint distribution of two random variables over the two sample sets and minimizing the loss in mutual information in going from individual samples to clusters of samples.

In our experiments in the citation deduplication domain, we are able to achieve results better than those achieved with coclustering. This is because we are able to take advantage of the coclustering structure with experts that take into account venues. In more complex domains, an extension of our approach to allow coclustering may be useful. One way to do this is to add a new root node that represents the joint clustering to the Bayesian network of Chapter 5. The prior probability distribution of this node would represent coclustering constraints such as those between venues and citations. Finding the optimal clustering in this new formulation becomes more difficult as the search would need to take into account the prior probabilities. We leave this extension as future work.

8.3.2 Clustering with Constraints

An alternative approach for combining experts is to interpret their output as *constraints* that can then be used to improve the performance of clustering algorithms. In this approach, experts would output their hypotheses as *must-link* or *cannot-link* constraints that respectively indicate whether two samples should or should not be in the same cluster[75]. A set of must-link and cannot-link constraints is *consistent* if there is at least one clustering

such that all constraints are satisfied.

The resulting set of constraints can then be used to improve clustering algorithms in a variety of ways. For example, constraints can be used to reduce the size of the clustering search space by assuring that the search only considers clusterings where all constraints are satisfied[76]. Alternatively, constraints can be used to distort the sample space such that samples referred in a must-link constraint are brought closer and samples referred in a cannot-link constraint are pushed farther apart[45]. Distorting the sample space has the advantage that the effects of constraints are propagated to samples that are in the neighborhood of the samples referred by the constraints. Assuming the constraints are *consistent*, another alternative is to use the constraints to generate initial partitionings. Clustering algorithms, such as k-means, that require a initial partitioning can benefit from generating the initial partition from constraints rather than randomly[4].

Constraints can also be used to train a classifier which can then extrapolate whether unseen pairs of samples are in the same cluster or not[18]. This approach is similar to our approach in that the clustering step uses a learned function between pairs rather than a pre-defined metric on the sample space.

8.3.3 Ensemble Clustering

Another research area that is relevant to our approach is ensemble clustering[70]. In ensemble clustering, the goal is to find a new clustering by combining the information in multiple clusterings of the same data set. This type of work focuses on defining a clustering criteria that takes into account how samples are clustered in different clusterings. This is relevant to our approach in that complete or partial clusterings rather than pairs can be used as the common language of experts. The combination of experts can then be done through ensemble clustering techniques. As a simplified example, consider taking advantage of both URL patterns and page content similarity to find clusters of pages. The pages can be independently clustered using URLs and page content, perhaps using a standard document clustering algorithm, and then the resulting clusterings can be combined using ensemble clustering techniques. The advantage of this is that if an expert does generate a clustering hypothesis as part of its processing, then the output captures the clustering hypothesis directly. With pairwise hints as the common language, the same clustering hypothesis needs to be expressed as a large collection of hypotheses on individual pairs.

8.4 Multi-Expert Approaches

Combining multiple experts, as we do for the web data extraction problem, has been applied to many difficult AI problems, such as document classification, scene interpretation[9] or speech recognition[30]. Under certain simplifying assumptions, such as prediction of 0-1 valued functions, it can be proven that it is possible to combine the output of multiple experts in a way so as to bound the number of errors as a function of the errors made by the *best* expert[8].

The theoretic results in combining multiple experts is typically done in the on-line learning setting[51]. Here, the meta-learner receives predictions from multiple learners, makes its prediction based solely on the predictions from the learners and then is given the correct prediction with which the meta-learner updates its model of the learners for its next prediction.

Even though the meta-learner may not perform as well as the best learner on particular instances, if there is not one particular learner that outperforms all others on all instances, then the meta-learning approach performs well because it is always bounded by the error rate of the best learner. This is also the intuition behind our approach: By combining multiple experts, we can achieve better extraction results on average than using particular types of structure.

When there are many experts to combine some of which are not relevant to the current problem instance, as is the case in text classification[21], it is more efficient to use only a subset of relevant experts[7, 32].

In *blackboard* systems[23], a number of experts collectively solve a problem by communicating through a blackboard. The common language in which messages are posted to and read from the blackboard is specifically designed for the type of problems that the system is built to solve. Our approach is similar to blackboard systems in that the experts' hypotheses are collected in a central location. Unlike blackboard systems, our experts do not read and make use of hypotheses posted by other experts, although this is an area that deserves future work: for example, experts that are computationally expensive are better run on small sub-problems as they arise from the hypotheses of cheaper experts. As a concrete example, it might not be possible to run an ISBN lookup on every ten digit number, but if some experts generate a hypothesis that a group of numbers are all ISBN numbers, the ISBN lookup expert could be run to verify this hypothesis.

Like blackboard systems, in multi-agent systems[28], multiple experts or *agents* are

utilized to solve problems. Unlike blackboard systems, the emphasis of multi-agent systems is to avoid central control of agents. Thus, multi-agent system research focuses building autonomous agents that can function without central control, and the communication(e.g., [43]) and coordination (e.g., [66]) of such autonomous agents.

Next we describe some example multi-expert systems and compare them to our approach.

8.4.1 GRAVA - Self-Adaptive Architecture

Robertson[62] describes a multi-agent architecture for solving vision problems robustly. He observes that scene interpretation can be solved relatively easily in specialized applications (such as in manufacturing) where the environment (lighting, sensor locations, objects in the scene, etc.) is well-known and tightly controlled, but the problem becomes much harder in applications where the environment changes dynamically. As a solution, Robertson proposes an adaptive and layered architecture, where higher layers interpret the output from lower layers. The interpretation provided by the top level is the output of the system. The architecture is adaptive because when unexpected results are found, a lower-layer can signal a higher-layer which then can regenerate the lower-layer using the additional knowledge that current assumptions are not valid.

In general, for a given problem instance, multiple interpretations are possible. GRAVA uses the MDL principle to pick the most likely interpretation. MDL provides a uniform measure across multiple agents even when the agents operate at differing levels of abstraction. Optimizing for MDL is equivalent to finding the most likely interpretation.

Tessar[73] describes an application of Robertson's architecture where the problem is to interpret ancient text. The two agents in this application are the character agent which matches the stroke information extracted from the image of the text to the characters in the training set and the word agent which matches the character sequences from the character agent against a corpus of words. Both agents compute the description lengths of their interpretations. For example, a perfect word match generates a description length that is only related to the frequency of the word in the corpus whereas a partial match generates a description length that combines the frequency of the word and the length of the description for a transformation that turns the word into the partially matched character sequence.

The application uses Monte Carlo approximation to search across interpretations. In the Monte Carlo approach, agents are asked to output interpretations multiple times. Each

agent picks an interpretation in relation to the description length of that interpretation: The shorter the description length, the more likely it is for an interpretation to be picked. The Monte Carlo approach allows the system to avoid local minima. If an agent always picks the most likely interpretation for its input, the overall most likely interpretation might never be found because it might be the case that the optimal high-level interpretation does not include the locally-optimal low-level interpretation. By running the Monte Carlo simulation many times, the distribution of the top-level interpretation is sampled. From this distribution, the most likely interpretation is picked as the solution.

Comparison to our approach

Robertson's approach uses MDL whereas we use probabilities to measure overall progress. The two measures can be shown to be equivalent under certain conditions. Regardless of whether they are equivalent or not, the two formalizations both provide a common unit of measure where values have a global meaning across heterogeneous experts and can be combined in a principled way.

Robertson has a separate interpreter that controls the activation of agents while we simply allow all experts to run on the problem. Robertson's approach works well when there is a clear hierarchy of experts such that the output of experts in one layer is consumed by experts at the next layer. Our approach avoids the additional complexity of an interpreter by having all experts run independent of each other. In simplifying the framework, we give up on the flexibility of having experts that use other experts' output.

8.4.2 Proverb - A Crossword Solver

Our approach of combining multiple experts in a probabilistic framework is similar to Proverb[67], a crossword puzzle solver. A *clue* in a crossword puzzle can have multiple answers. This is because there are multiple categories of clues such as synonyms, quotations, abbreviations, or geography and also because even when interpreted in a particular category, a clue can have multiple answers. One of the challenges of the crossword problem is that in finding the correct answer to a particular clue (in a particular crossword), one has to consider both multiple interpretations of the clue and the constraints of the answer grid.

Proverb has a number of "experts" that given a clue, output their list of best candidate answers together with probabilistic preferences assigned to each candidate. The solution

to the puzzle is found by globally optimizing the probability assignment for the particular choice of answers. Proverb's experts are analogous to our experts. Its common hypothesis language is lists of candidate answers with a confidence score attached to each answer.

8.4.3 Poirot - Integrated Learning

Poirot is an ambitious project that aims to achieve human-like performance in learning in planning domains. Given a single demonstration of a successful execution of a task, Poirot aims to learn to generate plans that solve similar tasks. For example, Poirot can be given the sequence of actions that a human user follows as he schedules a medical evacuation task and asked to solve another evacuation task but in the military domain instead of the civil domain.

Poirot follows a similar approach to ours at a high-level. It consists of a set of learners that generate hypotheses and a meta-learner that ultimately combines the hypotheses into one workflow. Learners are similar to our experts in that they work independently from each other, have access to use as much background knowledge as necessary, and output their hypotheses in LTML, the common language.

LTML is more expressive than our common language in general. Its expressions are in a higher-order logic, where hypotheses about other hypotheses can be expressed. In contrast, the common language we have experimented with only allows statements about pairs of samples. One area in which it is more expressive than LTML is on probabilistic statements. In our common language, each hypothesis has a confidence score associated with it whereas in LTML hypotheses are binary: they are either posted by the experts or not.

Having an arbitrarily complex "common" language makes combining the experts difficult. In our approach, the experts are combined simply by finding the most-likely clustering given the hypotheses about pairs from the experts. Poirot follows a different approach where some meta-experts examine posted hypotheses and post new ones to guide the learners so that eventually hypotheses describe a correct and complete workflow.

The termination of our approach is well defined. First experts generate hints, then the most likely model is found and returned as the solution. Poirot follows a more iterative approach. Experts continually read and post hypotheses until a meta-learner determines that enough hypotheses have been generated to form a solution.

Chapter 9

Conclusions and Future Work

9.1 Summary

In this dissertation, we introduced a new approach to solve the problem of web site extraction, which falls into the general class of structure discovery problems. Site extraction is the problem of finding a set of relational tables that best represent the data available on a web site. This is an important problem as the web contains enormous amounts of data which lays under a representation layer that facilitates human use but hides it from machines. Extensive research has been done on making it easier to extract data from the web and put it in machine-accessible form, but in general research has focused on extracting particular types of data or from particular sites or the approaches have required human effort for training. In this dissertation, we introduced a new approach of tackling the problem of site extraction. The new approach applies to many types of web sites and even other domains where the problem can be cast as a structure discovery problem. Our approach also reduces the the human effort involved in extracting data.

The main principle of the new approach is to take advantage of many different types of clues, such as URL patterns or visual layout, that appear on a web site. This is done by combining experts each of which focuses on a particular type of clue. Following this principle, we developed two frameworks.

In the first framework, our focus was on developing a common hypotheses language. A common language allows a collection of heterogeneous experts to express their discoveries in a uniform way and the output from the experts to be combined in a principled fashion. With the common language defined, we went on to develop a set of experts for the site

extraction problem. Our experiments with this system showed that it was able to achieve good extraction accuracy in terms of the standard precision, recall and F1 measures on a number of sites of different types, but it was also clear that better performance was still possible as the first framework has an inherent shortcoming: experts are not able to assign confidence scores to their hypotheses. Next, we developed a second framework where the common language allows confidence scores on hypotheses.

In the second framework, each expert not only generates hypotheses from the particular type of structure it specializes in, but also assign confidence scores to its hypotheses. This allows more information from the experts to be available while the hypotheses are combined. We experimented with the second framework first in the web domain where we improved on the results of the first framework. Then, we applied the same framework to the record linkage domain by developing a set of experts specific to this domain. This allowed us to do a comparative study against published results, which was difficult to do in the web domain. Our results indicate that combining experts that analyze multiple types of structure compares well to a framework that learns complex interdependencies between simple features.

9.2 Contributions

The main contribution of this work is a new approach to data extraction from the web. In our thesis statement we claimed that by encapsulating human knowledge about different types of web structure as software experts and combining these experts while maintaining a consistent model of the underlying data, we can solve the web data extraction problem successfully. To this end, we developed two frameworks for combining software experts, defined a cluster-based representation of data as an intermediate form from which the relational form can be derived, built experts that understand common types of web structure, and demonstrated that the approach achieves high accuracy in data extraction from the web.

Other contributions are:

- **Site Extraction**: We defined the problem of site extraction, which is the problem of finding the relational form of all data on a given web site. This is a novel way to look at the problem of data extraction from the web as previous work focuses on extracting specific types of data, such as person names, addresses, or dates, or

extracting data from specific page-types. Considering the site as a whole instead gives our experts a potentially richer set of structures to work on.

- **Clustering with Heterogeneous Experts**: We introduced a new approach of combining heterogeneous experts to solve clustering problems. The approach relies on a common language in which experts output their findings and a search strategy that ensures global consistency of the solution. In particular, the common language requires each hypothesis to assert whether a pair of samples are in the same cluster or not. The common language allows our approach to be able to make use of the output from any expert regardless of the particular type of structure that it is working on. An important aspect of the process of combining experts is that global consistency of the data model is maintained. For the clustering approach, this means that the solution is a proper partitioning of the samples so that each sample belongs to one and exactly one cluster.
- **Implementation**: We implemented a flexible clustering system into which any number of experts can be added. We also developed a number of sophisticated experts that understand complicated web structures. The clustering system and the web experts can be used together for the site extraction problem. The clustering system and the set of web experts are also useful on their own for future work.

9.3 Future work

9.3.1 Theoretical Framework

In this work, we viewed the problem of data extraction from the web as a structure discovery problem, tackled it by combining heterogeneous experts that encapsulate human knowledge about surface structure and showed empirically that the approach works in the web and record linkage domains. An important next step is to build a theoretical framework of structure discovery via experts that understand surface structure. A theory of structure discovery would define hidden and surface "structure" formally, characterize the relation between hidden and surface structure, characterize experts in terms of how accurately they represent this relation, all with the goal of tying together coverage and accuracy of experts, complexity of the hidden structure and the characteristics of the relation of hidden and surface structure.

9.3.2 Representation of Underlying Structure

The two frameworks we developed use a clustered-based representation as an intermediate form from which the ultimate relational representation can be derived. Alternatively, the data can be directly modeled as relational tables rather than as clusters. This would potentially allow experts to be more explicit in their discoveries. For example, the page layout expert can generate separate hypotheses about data in rows versus data in columns, assuming that the horizontal versus vertical layout of data is reflective about the underlying structure of data. The challenge in doing this is to develop efficient search algorithms that can search through the space of such models.

9.3.3 Iterative Interpretation

Perhaps a more powerful approach to structure discovery is an iterative approach where at each iteration, the model of the underlying data structure is updated based on the current set of hypotheses from the experts' and then experts re-"interpret" the problem while taking into account the current model. This is similar to blackboard systems as discussed earlier, with the exception that the experts consider only the current model rather than hypotheses that have been generated by other experts. The current work does not directly address this kind of iterative reasoning. An interesting path for future work is investigating how experts can be made to make use of this type of dynamic knowledge.

9.3.4 Other Domains

To further validate the principles of our approach, it will be necessary to apply it in other domains. This involves three areas of work:

- **Developing Experts**: One of the challenges in applying our approach to new domains is developing the set of experts which understand the common types of structure particular to that domain. This is a more difficult task the similar task of defining features. Features allow samples to be represented in a simple, uniform manner so that learning and using more complicated structures of the domain remain as part of the discovery or learning problem. In contrast, experts encode as much as possible of the known structures so that the discovery process can make use of existing knowledge.

- **Common Language**: Another task in applying our approach to new domains is the selection of a common language. The common language needs to be chosen with the experts and the representation of the underlying structure in mind. A good common language allows experts that work on heterogeneous structures to express their discoveries naturally, but also is not overly complex that the search for the underlying structure becomes prohibitive.
- **Search**: In our approach, we used a cluster-based representation of the underlying data. This allowed us to use standard clustering algorithms, such as the leader-follower algorithm and the agglomerative clustering algorithm, as our "search" algorithm. However, other data representations will require new search strategies. Consider the problem of scene-recognition where the hidden 3D-model consists of known set of solid objects. The search space for this problem is the set of all configurations in 3D-space of the known objects such that none of the physical-world constraints are violated. Effectively searching this space to find the solution that maximally agrees with the hypotheses requires a strategy specific to this search space and the hypothesis language.

Bibliography

[1] Edoardo M. Airoldi, William W. Cohen, and Stephen E. Fienberg. Bayesian methods for frequent terms in text: Models of contagion and the Δ^2 statistic. In *CSNA & INTERFACE Annual Meetings*, St. Louis, MI, 2005. 3.4

[2] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study, 1998. 4.1.2

[3] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, pages 337–348, 2003. 2.1

[4] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD04*, pages 59–68, Seattle, WA, August 2004. 8.3.2

[5] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, New York, NY, USA, 2003. ACM Press. 7.1

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. 3.4

[7] Avrim Blum. Empirical support for Winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In *Proc. 12th International Conference on Machine Learning*, pages 64–72, San Francisco, CA., 1995. Morgan Kaufmann. 8.4

[8] Avrim Blum. On-line algorithms in machine learning. In *Online Algorithms*, pages 306–325, 1996. 8.4

[9] W. Butera and V. Bove. The coding ecology: Image coding via competition among experts, 2000. 8.4

[10] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, pages 6–11, Menlo Park, CA, 1998. AAAI Press. 8.2

[11] CALO. http://www.ai.sri.com/project/calo. 4.2

[12] Chia-Hui Chang, Moheb Ramzy Girgis, Mohammed Kayed, and Khaled Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006. 8.1.1

[13] H. Chen, S. Tsai, and J. Tsai. Mining tables from large scale html texts, 2000. 8.1.2

[14] Yejin Choi and Claire Cardie. Structured local training and biased potential functions for conditional random fields with application to coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 65–72, Rochester, New York, April 2007. Association for Computational Linguistics. 6.2.1

[15] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970. 3.1

[16] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. 3.7

[17] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks, 2003. 3.7

[18] W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration, 2002. 8.3.2

[19] William W. Cohen and Wei Fan. Learning page-independent heuristics for extracting data from web pages. In *In AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999. 8.1.1

[20] William W. Cohen, Matthew Hurst, and Lee S. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, pages 232–241. ACM Press, 2002. 8.1.1

[21] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text

categorization. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 307–315, Zürich, CH, 1996. ACM Press, New York, US. 8.4

[22] Diane J. Cook and Lawrence B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000. 8.2.1

[23] Daniel D Corkill. Collaborating Software: Blackboard and Multi-Agent Systems & the Future. In *Proceedings of the International Lisp Conference*, New York, New York, October 2003. 8.4

[24] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001. 2.1, 8.1.3

[25] Aron Culotta and Andrew McCallum. Joint deduplication of multiple record types in relational data. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 257–258, New York, NY, USA, 2005. ACM Press. 3.2, 5.4.1, 6.2.1, 7.1, 8.3.1

[26] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98, 2003. 5.4.1

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000. 3.2, 3.4, 3.5, 8.3

[28] E.H. Durfee, V.R. Lesser, and D.D. Corkill. Distributed Problem Solving. *The Encyclopedia of Artificial Intelligence, Second Edition*, January 1991. 8.4

[29] David W. Embley, Cui Tao, and Stephen W. Liddle. Automatically extracting ontologically specified data from html tables of unknown structure. In *ER '02: Proceedings of the 21st International Conference on Conceptual Modeling*, pages 322–337. Springer-Verlag, 2002. 8.1.2

[30] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213–253, June 1980. 8.4

[31] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the*

American Statistical Association, 64(328):1183–1210, 1969. 7.1

[32] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. pages 334–343, 1997. 8.4

[33] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 71–80, New York, NY, USA, 2007. ACM. 8.1.2

[34] Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of relational structure. In *Proc. 18th International Conf. on Machine Learning*, pages 170–177. Morgan Kaufmann, San Francisco, CA, 2001. 8.2

[35] Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification, 2001. IJCAI Workshop on "Text Learning: Beyond Supervision". 8.2

[36] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, New York, NY, USA, 2005. ACM Press. 1.1.1

[37] Michael Habeck, Michael Nilges, and Wolfgang Rieping. Bayesian inference applied to macromolecular structure determination. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72(3):031912, 2005. 2.2

[38] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, 2007. 1.1.1

[39] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999. 3.4

[40] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):1–14, 1997. 3.2

[41] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. 8.3

[42] Hailin Jin, Anthony J. Yezzi, and Stefano Soatto. Mumford-shah on the move: Region-based segmentation on deforming manifolds with application to 3-d reconstruction of shape and appearance from multi-view images. *J. Math. Imaging Vis.*, 29(2-3):219–234, 2007. 2.2

[43] Hyuckchul Jung and Milind Tambe. On communication in distributed constraint satisfaction. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1480–1481, Washington, DC, USA, 2004. IEEE Computer Society. 8.4

[44] Zu Whan Kim and Ramakant Nevatia. Expandable bayesian networks for 3d object description from multiple views and multiple mode inputs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(6):769–774, 2003. 5.2.1

[45] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In Claude Sammut and Achim G. Hoffmann, editors, *ICML*, pages 307–314. Morgan Kaufmann, 2002. 8.3.2

[46] Craig A. Knoblock, Kristina Lerman, Steven Minton, and Ion Muslea. *Accurately and reliably extracting data from the web: A machine learning approach*, pages 275–287. Intelligent Exploration of the Web. Springer-Verlag, Berkeley, CA, 2003. 1.1.2, 4.1.1, 8.1.1

[47] Bernhard Krüpl, Marcus Herzog, and Wolfgang Gatterbauer. Using visual cues for extraction of tabular data from arbitrary HTML documents. In *Proceedings of the special interest tracks and posters of the 14th international conference on World Wide Web (WWW 2005*, pages 1000–1001. ACM Press, May 2005. 8.1.2

[48] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000. 1.1.2, 8.1.1

[49] Kristina Lerman, Lise Getoor, Steven Minton, and Craig Knoblock. Using the structure of web sites for automatic segmentation of tables. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 119–130. ACM Press, 2004. 8.1.2

[50] Kristina Lerman, Steven Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *JAIR*, 18:149–181, 2003. 2.1

[51] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, 1988. 8.4

[52] Yan Liu, Jaime G. Carbonell, Peter Weigele, and Vanathi Gopalakrishnan. Segmentation conditional random fields (scrfs): A new approach for protein fold recognition. In Satoru Miyano, Jill P. Mesirov, Simon Kasif, Sorin Istrail, Pavel A. Pevzner, and

Michael S. Waterman, editors, *RECOMB*, volume 3500 of *Lecture Notes in Computer Science*, pages 408–422. Springer, 2005. 2.2

[53] Laszlo Lovasz. *Combinatorial Problems and Exercises*. Elsevier Science Ltd, 1979. 1

[54] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture, 2002. 2.3

[55] Steven Minton, Sorinel I. Ticrea, and Jennifer Beach. Trainability: Developing a responsive learning system. In *IIWeb*, pages 27–32, 2003. 4.2

[56] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US. 3.4

[57] Parag and P. Domingos. Multi-relational record linkage. In *Proceedings of 3rd Workshop on Multi-Relational Data Mining at ACM SIGKDD*, Seattle, WA, August 2004. 7.1

[58] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. 5.2.1

[59] A. Pivk, P. Cimiano, and Y. Sure. From tables to frames, 2005. 8.1.2

[60] Katharina Probst, Rayid Ghani, Marko Krema, Andrew E. Fano, and Yan Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In Manuela M. Veloso, editor, *IJCAI*, pages 2838–2843, 2007. 8.1.4

[61] P. Ravikumar and W. Cohen. A hierarchical graphical model for record linkage, 2004. 7.1

[62] Paul Robertson and Robert Laddaga. The grava self-adaptive architecture: History; design; applications; and challenges. In *ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04)*, pages 298–303, Washington, DC, USA, 2004. IEEE Computer Society. 8.4.1

[63] Gian-Carlo Rota. The number of partitions of a set. *American Mathematical Monthly*, 71(5):498–504, 1964. 1

[64] Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*.

McGraw-Hill Book Company, 1984. 3.4

[65] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986. 1.1.2

[66] P. Scerri, R. Vincent, and R. Mailler. Comparing three approaches to large scale coordination, 2004. 8.4

[67] Noam M. Shazeer, Michael L. Littman, and Greg A. Keim. Solving crossword puzzles as probabilistic constraint satisfaction. In *AAAI/IAAI*, pages 156–162, 1999. 8.4.2

[68] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, December 1996. 1.1.2

[69] Yang Song, Jian Huang, Isaac G. Councill, Jia Li, and C. Lee Giles. Efficient topic-based unsupervised name disambiguation. In *JCDL '07: Proceedings of the 2007 conference on Digital libraries*, pages 342–351, New York, NY, USA, 2007. ACM. 6.2.1

[70] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proc. Conference on Artificial Intelligence (AAAI 2002), Edmonton*, pages 93–98. AAAI/MIT Press, July 2002. 8.3.3

[71] Benjamin Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In Bernhard Nebel, editor, *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, pages 870–878, Seattle, US, 2001. 8.2

[72] Ashwin Tengli, Yiming Yang, and Nian Li Ma. Learning table extraction from examples. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 987, Morristown, NJ, USA, 2004. Association for Computational Linguistics. 8.1.2

[73] Melissa M. Terras. *Image to Interpretation*. Oxford, 2006. 8.4.1

[74] Kentaro Toyama and Eric Horvitz. Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *Proceedings of ACCV '00, Fourth Asian Conference on Computer Vision*, 2000. 5.2.1

[75] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*,

pages 1103–1110, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 8.3.2

[76] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 8.3.2

[77] H. L. Wang, S. H. Wu, I. C. Wang, C. L. Sung, W. L. Hsu, and W. K. Shih. Semantic search on internet tabular information extraction for answering queries. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 243–249, New York, NY, USA, 2000. ACM. 8.1.2

[78] G. I. Webb and D. Brain. Generality is predictive of predication accuracy. In T. Yamaguchi, A. Hoffmann, H. Motoda, and P. Compton, editors, *Proceedings of the 2002 Pacific Rim Knowledge Acquisition Workshop (PKAW'02)*, pages 117–130, Tokyo, 2002. Japanese Society for Artificial Intelligence. 2.3

[79] W. Winkler. The state of record linkage and current research problems, 1999. 2.2

[80] Tak-Lam Wong, Wai Lam, and Tik-Shun Wong. An unsupervised framework for extracting and normalizing product attributes from multiple web sites. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42, New York, NY, USA, 2008. ACM. 8.1.4

[81] R. Zanibbi, D. Blostein, and J.R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. 8.1.2

[82] Yanhong Zhai and Bing Liu. Automatic wrapper generation using tree matching and partial tree alignment. In *AAAI*. AAAI Press, 2006. 8.1.3