Coupled Semi-Supervised Learning

Andrew Carlson

May 2010
CMU-ML-10-104

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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

This research was sponsored by a Yahoo! PhD Student Fellowship, a grant from Google, the Air Force Research Laboratory under grant numbers FA8750-08-1-0009 and FA8750-09-C-0179, and SRI International under grant number 03000211. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Machine learning, semi-supervised learning, coupled semi-supervised learning, information extraction, web mining, pattern learning, wrapper induction

Abstract

This thesis argues that successful semi-supervised learning is improved by learning many functions at once in a *coupled* manner. Given knowledge about constraints between functions to be learned (e.g., $f_1(x) \rightarrow \neg f_2(x)$), forcing the models that are learned to obey these constraints can yield a more constrained, and therefore easier, set of learning problems. We apply these ideas to bootstrap learning methods as well as semi-supervised logistic regression models, and show that considerable improvements are achieved in both settings. In experimental work, we focus on the problem of extracting factual knowledge from the web. This problem is an ideal case study for the general problems that we study because there is an abundance of unlabeled web page data available, and because thousands or millions of functions are discussed on the web.

Chapter 3 focuses on coupling the semi-supervised learning of information extractors that extract information from free text using textual extraction patterns (e.g., "mayor of X" and "Y star quarterback X"). We present an approach in which the input to the learner is an ontology defining a set of target categories and relations to be learned, a handful of seed examples for each, and a set of constraints that couple the various categories and relations (e.g., Person and Sport are mutually exclusive). We show that given this input and millions of unlabeled documents, a semi-supervised learning procedure can, by avoiding violations of the constraints in how its learned extractors label unlabeled data, achieve very significant accuracy improvements over semi-supervised methods that do not avoid such violations.

In Chapter 4, we apply the ideas from Chapter 3 to a different type of extraction method, wrapper induction for semi-structured web pages. We also consider how to couple multiple extraction

methods that typically make independent errors. To couple pattern-based extraction and wrapper-based extraction, we use a strategy that only promotes instances extracted by both methods. Experimental results on dozens of categories and relations demonstrate that coupling wrapper induction improves the precision of the promoted facts, and that coupling multiple extraction methods leads to higher precision than either of the methods alone.

In Chapter 5, we consider two questions: (1) Can we scale up the number and variety of predicates in our ontology and still maintain high precision with coupled semi-supervised learning methods? and (2) Should we consider adding additional extraction methods beyond textual patterns and wrappers? We first describe a general architecture that can exploit many different extraction methods. We then describe a prototype implementation of our architecture, called *Multi-Extractor Coupler* (MEC). With an extended ontology of 123 categories and 55 relations, MEC has learned to extract a knowledge base containing over 242,000 beliefs with an estimated precision of 74%.

Chapter 6 considers how to couple the semi-supervised learning of logistic regression models. Specifically, we consider learning many binary logistic regression classifiers when many pairs of classes are known to be mutually exclusive. We present a method that uses unlabeled data through a penalty function that regularizes the training of classifiers by penalizing violations of mutual exclusion. We apply this idea to training classifiers which decide if a noun phrase is a member of some specific category. Semi-supervised training of such classifiers is shown to improve performance relative to supervised-only training. We speculate that use of similar penalty functions could provide an alternative to the methods for coupled semi-supervised learning presented in previous chapters, with the advantage that the models being learned are principled, probablistic models that are easy to train and can be applied to any example to provide a prediction of posterior probabilities.

For Julia

Acknowledgments

I would first like to thank my funders. This work was supported in part by the Air Force Research Laboratory, SRI International, Google, and a Yahoo! Fellowship. The fellowship in particular allowed me freedom to be ambitious with this research. Yahoo! also must be acknowledged for generously providing the M45 computing cluster, without which the web-scale nature of this work would have been impossible. Not only did Yahoo! provide this great resource, but it was impressively well administered, too.

Jamie Callan and his research group provided a first 200-million web page crawl, and then later officially released the ClueWeb09 data set with 500 million English-language web pages. This saved our research group months of effort.

Carnegie Mellon deserves recognition for starting the first Machine Learning Department in the world. I am grateful to the professors in the MLD from whom I learned so much in courses. I am also grateful that the MLD let Tom Mitchell teach a unique and open-ended course called "Read the Web" that grew into our research effort.

I would like to thank my committee for providing comments and suggestions that improved the thesis in many ways and places, and for taking time out of their busy lives to help me.

My fellow students at Carnegie Mellon provided countless insights and helpful conversations, feedback on drafts and practice talks, and a community of interesting and wildly varied people, too. The willingness of students at CMU to chat about one another's research is something special.

Thank you to my colleagues in the Read the Web research group: Bryan, Burr, Estevam, Jayant, Justin, Richard, Sophie, Sue Ann, Tom, and Weam. It has been such a pleasure working on this project with you, and I look forward to seeing where it goes next!

I would like to thank Diane Stidle for years of help with countless things (from visiting Pittsburgh to figuring out requirements to figuring out how to graduate). The department would fall apart without you, Diane.

Sharon Cavlovich, for years of help booking flights, helping with meetings, and dozens of other things (including helping Julia and I understand the meaning of Graham's cries courtesy of a DVD).

Had Dan Roth not taken me on as an undergraduate researcher at the University of Illinois at Urbana-Champaign, I probably would not have discovered the great field of Machine Learning. Thanks, Dan, for helping me find my passion, and for taking the time to nuture undergraduates as they discover research.

Tom Mitchell, for wisdom, patience, insight, and also being among the nicest people around. It has been an incredible pleasure to work with and learn from you. Thank you for believing in this project from the start, and thank you for treating me like a colleague from day one.

To my parents, without whose sacrifice none of this would have been possible, and to my family, who provided love and support, and in particular supported my tinkering with computers from the IBM PCjr in the basement at home all the way through college.

And finally, to Julia, whose patience, love, encouragement, and support were essential. Without you, I never would have started my PhD and I never could have finished, either. Thank you for helping me through these past five years, and for making sacrifices to make it all possible. And to Graham, with whom I feel immensely blessed, and whose generous naps made it possible to finish this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Great successes have been achieved with supervised machine learning methods. Such methods involve training a model from a pool of labeled examples of some function to be learned. Such methods are now the standard approach to a variety of problems where explicitly writing down a program is not feasible but learning one from data is, such as speech recognition, machine translation, and classification of medical images. However, the practical application of machine learning methods is greatly constrained by the lack of availability of large labeled training sets, and the effort required to create them. This motivates the development of semi-supervised methods. Semi-supervised methods exploit unlabeled data in addition to labeled data to learn models. When successful, less labeled data is needed to reach a given level of performance using semi-supervised methods rather than purely supervised methods.

Some of the earliest successes in semi-supervised learning came from bootstrap learning methods (also called self-training or self-supervised methods) [Yarowsky, 1995; Blum and Mitchell, 1998; Collins and Singer, 1999]. Bootstrap learning methods train on the available labeled data, use the current model to label some of the unlabeled data, train on the newly expanded labeled data, and repeat. Such methods have shown promise, but suffer from issues of semantic drift, where errors in the newly self-labeled data could cause the system to run off track [Curran *et al.*, 2007]. If methods could be developed to mitigate this drift, it could yield a significant step forward for semi-supervised learning.

One approach to forestalling drift is learning several mutually exclusive functions together [Riloff and Jones, 1999; Yangarber, 2003; Etzioni *et al.*, 2005]. The idea is that if different classes are mutually exclusive, then the classes can constrain each other through this relationship. Empirical results demonstrated significantly higher accuracies using these methods, but drift still occurs.

Chapters 3, 4, and 5 of this thesis explore the idea that learning many different functions with different types of constraints in addition to the mutual exclusion constraints will enable more accurate bootstrap learning.

Another type of semi-supervised machine learning involves using unlabeled data and domain knowledge to regularize the training of exponential models. Exponential models, such as logistic regression and conditional random fields, are typically trained by maximizing an objective function that sums conditional likelihood with a regularization penalty to encourage small weights. Semi-supervised methods have added an additional term, which penalizes undesired behavior on unlabeled data. Penalties might enforce that the learned model matches known class priors [Mann and McCallum, 2007], or that the model tends to predict a certain class when a certain feature is active (e.g., the class “Hockey” when the word “puck” is in a document) [Druck *et al.*, 2008].

Chapter 6 of this thesis explores how to learn many binary logistic regression models and couple their semi-supervised training by exploiting the knowledge that certain pairs of classes being learned are mutually exclusive by adding a regularization term that penalizes violations of those mutual exclusion relationships to the objective function.

1.2 Thesis Statement

The central thesis of this work is that it is possible to learn many accurate function approximators with relatively little labeled data by leveraging semi-supervised learning methods, a large pool of unlabeled data, and by using relationships that exist between the functions being learned to couple the learning of those functions.

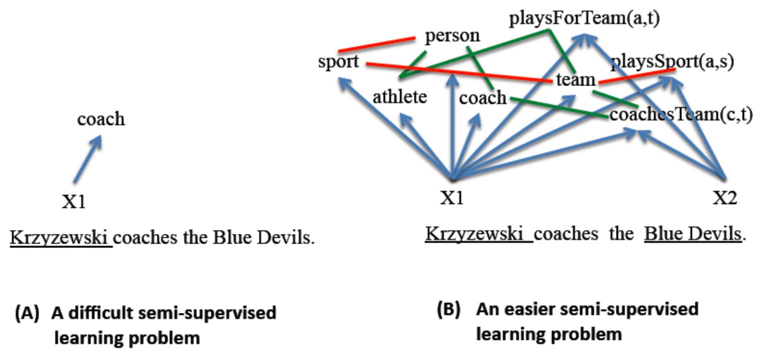

coach
X1
Krzyzewski coaches the Blue Devils.
(A) A difficult semi-supervised learning problem
playsForTeam(a,t)
person
sport
playsSport(a,s)
team
athlete
coach
coachesTeam(c,t)
X1
X2
Krzyzewski coaches the Blue Devils.
(B) An easier semi-supervised learning problem

Figure 1.1: This thesis argues that in semi-supervised learning, coupling the training of many classifiers for entities and relations (B) results in a more constrained, and therefore easier learning problem than training a single classifier (A).

This idea is illustrated in Figure 1.1. Here arrows represent the functions being learned (specifically, labeling a noun phrase as to whether or not it refer to an instance of a category like "coach", and labeling pairs of noun phrases as to whether or not some relation like "playsForTeam" holds between them), and lines represent constraints between the outputs of the functions being learned (e.g., "athlete" and "team" are the argument types of "playsForTeam"). At first glance, problem (B) seems like the harder learning problem, because it looks much more complex. This thesis argues that problem (B) is the easier semi-supervised learning problem, because it is more constrained.

More formally, assume that we have some instance space $\mathcal{X}$. We are learning a collection of n functions $f_1, f_2, \ldots, f_n$. $f_i : \mathcal{X} \rightarrow \mathcal{Y}_i$, where $\mathcal{Y}_i$ is some function-specific output space. Assume that we also have m constraint functions $\chi_1, \chi_2, \ldots, \chi_m$ that map from a vector of function outputs to a binary output. That is, $\chi_i : \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_m \to \{0, 1\}$. These constraint functions put constraints on the outputs of the functions so that only some function outputs are compatible.

Assume that data are generated from some data distribution $\mathcal{D}$ (the join distribution over $X, Y_1, Y_2, \ldots, Y_n$). We further assume that the data distribution $\mathcal{D}$ only puts positive probability on function outputs where all constraints are obeyed (signified by all constraint functions outputting a value of 1). We are given a set of labeled pairs (x, y_i) for each function f_i. We also have a collection of unlabeled examples. The goal is to learn accurate approximations of each function.

This class of problems is important because there are many real-world problems with many functions to be learned and obvious constraints to use where not much labeled data is available for each function and unlabeled data is abundant.

1.3 Case Study: Web Information Extraction

In experimental work, this thesis focuses on extracting facts about entities[1] and relations between entities from web text. This domain serves as an ideal case study for this work for several reasons:

- A vast collection (i.e., billions of documents) of unlabeled text that discusses entities and relations between them exists on the web.

- This problem domain can be modeled using thousands of functions (e.g. City($\cdot$), IsMayorOf($\cdot, \cdot$)) and a rich network of constraints between them. Some examples of such constraints include:

  - $\mathit{City}(x) \to \mathit{Location}(x)$
  - $\mathit{IsMayorOf}(x, y) \to \mathit{City}(x) \land \mathit{Person}(y)$
  - $\mathit{City}(x) \to \neg \mathit{Country}(x)$

[1]Entities are represented using strings; we do not resolve synonymous strings to real-world entities in this work.

- The extracted knowledge from this work will be valuable to many downstream applications.

As an example of how to fit this problem into our framework, assume that our instance space $\mathcal{X}$ is a set of pairs of some smaller space X, where the smaller space X describes noun phrases that could potentially refer to entities. Let $f_1(x_1, x_2)$ be the Person(x_1) predicate (which ignores the value of x_2), so that $f_1(x_1, x_2)$ is 1 if x_1 refers to a person, and 0 otherwise. Let $f_2(x_1, x_2)$ be the City(x_2) predicate, so that $f_2(x_1, x_2)$ is 1 if x_2 refers to a city, and 0 otherwise. Finally, $f_3(x_1, x_2)$ the IsMayorOf(x_1, x_2) predicate, with value 1 if x_1 refers to the mayor of a city referred to by x_2, and 0 otherwise. We could perform type checking of f_3 by defining two constraint functions:

- $\chi_1(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2))$ is defined to be 1 if $f_3(x_1, x_2) \rightarrow f_1(x_1, x_2)$ is satisfied, and 0 otherwise.

- $\chi_2(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2))$ is defined to be 1 if $f_3(x_1, x_2) \rightarrow f_2(x_1, x_2)$ is satisfied, and 0 otherwise.

1.4 Supplementary Online Materials

Numerous materials from this thesis have been put online at `http://rtw.ml.cmu.edu/acarlson_thesis`.

1.4.1 Ontologies

Two ontologies are available online. The first, smaller one was used in the experiments in Chapters 3 and 4, and the larger one was used in Chapters 5 and 6.

For categories, ontologies contain the following information:

- The parent category in the hierarchy of categories

- Exceptions to mutual exclusion (all pairs of categories are assumed to be mutually exclusive unless specified as exceptions or unless one is an ancestor of the other)
- Whether or not instances of the category are proper nouns, common nouns, or either
- Seed instances of the category
- Whether or not the system should try to populate the category (mostly "true" except for a few categories like Continent)
- Seed extraction patterns

For relations, ontologies contain the following information:

- The categories that are the domain and range of the relation (to enforce type checking)
- Whether or not the relation is symmetric (MBL in Chapter 4 and the Knowledge Integrator in Chapter 5 infer positive examples by swapping arguments if this is "true")
- Whether or not the relation is antisymmetric (similarly to symmetric, but used to infer negative examples)
- Negative examples of the relation
- Exceptions to mutual exclusion (all relations are assumed to be mutually exclusive unless specified as exceptions or unless one is an ancestor of the other)
- Seed instances of the relation
- The arity of the relation (one value vs. many) (if "1", only one value will be promoted for any given value in the domain of the relation)

1.4.2 Lists Used in Segmenting Noun Phrases

As described in Section 3.4.2 under "Category Instances," part-of-speech tags are used to detect and segment noun phrases in free text. To properly attach common prefixes (e.g., "St" and "Prof") and suffixes (e.g., “Sr” and “Inc”) to noun phrases, lists of each were provided to the system. These lists are available online. Also, noun phrases (as detected by the system) are ignored if they do not contain a word that is not on a list of English stop words. This list is also available online. Finally, proper noun phrases that contain prepositions or conjunctions are segmented using a list of phrases that was generated using a reimplementation of the Lex algorithm [Downey *et al.*, 2007]. This list is posted online.

1.4.3 Other Materials

As much as possible, results from the experiments in each chapter have been posted online. The specific materials posted are described toward the end of each chapter. These resources will hopefully be useful as a source of knowledge for future researchers. For example, the list of nearly a quarter-million facts learned by MEC in Chapter 5 could be useful as a semantic resource, while the extraction patterns learned by CPL in Chapter 3 could be used to bootstrap an information-extraction system that annotates individual sentences.

1.5 Overview of the Thesis

1.5.1 Survey of Related Work

Chapter 2 surveys work related to this thesis. First, semi-supervised learning is discussed, focusing on the sub-areas most relevant to this thesis: bootstrap learning, semi-supervised exponential models, and frameworks for semi-supervised learning. Second, we give an overview of techniques for extracting factual information from the web. Finally, we discuss methods for multi-task machine learning.

1.5.2 Coupled Learning of Textual Extraction Patterns

In Chapter 3, we focus on methods of coupling the semi-supervised learning of information extractors that extract information from free text using textual extraction patterns (e.g., "mayor of X" and "Y star quarterback X"). We identify three general types of coupling among target functions that can be combined to form a dense network of coupled learning problems. We then present an approach in which the input to the learner is an ontology defining a set of target categories and relations to be learned, a handful of seed examples for each, and a set of constraints that couple the various categories and relations (e.g., Person and Sport are mutually exclusive). We show that given this input and millions of unlabeled documents, a semi-supervised learning procedure can achieve very significant accuracy improvements by coupling the training of textual pattern-based extractors for dozens of categories and relations. Based on results reported here, we hypothesize that even greater accuracy improvements will be possible by forming a larger and more dense network of inter-constrained learning tasks. The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous semi-supervised training of both category and relation textual pattern-based extractors and (2) this work proposes that learning many tasks and coupling them as much as possible leads to higher accuracy semi-supervised learning, and provides web-scale experimental evidence to support that point.

1.5.3 Coupling Wrapper Induction and Multiple Extractors

In Chapter 4, we first consider applying the ideas presented in the previous chapter for coupling pattern-based information extraction to a different type of extraction method, wrapper induction for semi-structured web pages. We explore using coupling constraints based on mutual exclusion and type checking of relation arguments to learn more accurate wrappers within a bootstrap learning process. We then consider how to couple multiple extraction methods that typically make independent errors: the method of pattern-based extraction from unstructured text performed in the previous chapter and the method for wrapper-based extraction from semi-structured documents which we discuss first in this chapter. To couple these two methods, we use a strategy that only promotes instances extracted by both methods. Experimental results on dozens of categories and relations demonstrate that coupling wrapper induction improves the precision of the promoted facts, and that coupling multiple extraction methods leads to higher precision than either of the methods alone. The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous training of multiple wrapper inducers (2) this work is the first to couple the simultaneous training of multiple extraction methods (rather than simply combining the output of multiple extraction methods after training).

1.5.4 Scaling Up: More Predicates, More Extraction Methods

The results in previous chapters lead to two questions: (1) Can we scale up the number and variety of predicates in our ontology to over 100 categories and 50 relations and still maintain high precision with coupled semi-supervised learning methods? and (2) Should we consider adding additional extraction methods to the two methods used in the previous two chapters, coupling more than two techniques together? In Chapter 5, we explore these questions by learning to extract over 150 predicates, and by using four different extraction methods. We first describe a general architecture that can exploit many different extraction methods. The architecture uses coupled semi-supervised learning methods, an ensemble of varied knowledge extraction methods, and a flexible knowledge base that allows the integration of the outputs of those methods. We also discuss design principles for implementing this architecture. We then describe a prototype implementation of our architecture, called *Multi-Extractor Coupler* (MEC). With an extended ontology of 123 categories and 55 relations, MEC has learned to extract a knowledge base containing over 242,000 beliefs with an estimated precision of 74%. Analysis of the results shows that each of the four extraction methods contributes positively to these results.

1.5.5 Coupled Semi-Supervised Logistic Regression

In Chapter 6, we consider how to couple the semi-supervised learning of logistic regression models. Specifically, we consider learning many binary logistic regression classifiers when many (but not all) pairs of classes are known to be mutually exclusive. We present a method that uses unlabeled data through a penalty function that regularizes the training of classifiers by penalizing violations of mutual exclusion. We then apply this idea to training classifiers which decide if a noun phrase is a member of some specific category. Semi-supervised training of such classifiers is shown to improve performance relative to supervised-only training, and to slightly improve performance relative to the Coupled Pattern Learner method presented in Chapter 3. We speculate that use of similar penalty functions could provide an alternative to the methods for coupled semi-supervised learning presented in previous chapters, with the advantage that the models being learned are principled, probablistic models that are easy to train and can be applied to any noun phrase.

Chapter 2

Survey of Related Work

This chapter provides a review of work that is related to this thesis. The thesis is related to work on a number of different active research topics, each of which are discussed below. First, semi-supervised learning is discussed, focusing on the sub-areas most relevant to this thesis: bootstrap learning, semi-supervised exponential models, and frameworks for semi-supervised learning. Second, we give an overview of techniques for extracting factual information from the web. Finally, we discuss methods for multi-task machine learning.

2.1 Semi-Supervised Learning

This thesis aims to learn from a small amount of labeled data and a large amount of unlabeled data. This problem setting is called *semi-supervised learning*. A good overview of semi-supervised learning is provided by Zhu [2008]. Several different families of semi-supervised learning methods have been developed. For example, the EM algorithm can be used to deal with missing data in probabilistic modeling settings [Dempster *et al.*, 1977]. Nigam *et al.* [2006] use this technique to learn text classifiers and show that using unlabeled data improvs accuracy. Other techniques assume that decision boundaries of classifiers should go through low-density regions of the input space (e.g., Transductive Support Vector Machines [Vapnik, 1998]). Graph-based methods use a graph structure that connects examples that are similar in the input space and assume that examples that are close on the graph should have similar predictions [Zhu *et al.*, 2003].

There are two specific families of semi-supervised learning techniques that are used in this thesis. Both are discussed in detail below. The first, bootstrap learning, uses algorithms where a model is learned from initial labeled data and then used to label additional data. The second, regularized exponential models, uses traditional exponential models and supplements the objective function with a regularizer that encourages desired behaviors on unlabeled data.

Finally, we will discuss some theoretical frameworks for semi-supervised learning that incorporate human-specified constraints.

2.1.1 Bootstrap Learning

A large portion of this thesis uses bootstrap learning approaches to semi-supervised learning.[1] Bootstrap learning approaches[2] start with a small number of labeled "seed" examples of the class to be extracted, use those seed examples to train an initial model, then use this trained model to label some of the unlabeled data. A new model is then trained, using labeled data consisting of the original seed examples plus the new self-labeled examples. This process iterates, gradually expanding the amount of labeled data. Such approaches have shown promise in applications such as word sense disambiguation [Yarowsky, 1995], web page classification [Blum and Mitchell, 1998], named entity classification [Collins and Singer, 1999], parsing [McClosky *et al.*, 2006], and machine translation [Ueffing, 2006].

[1]The name "bootstrap learning" derives from the idea that such methods *bootstrap* themselves from a small amount of labeled data. These methods have nothing in common with the bootstrap technique in statistics.

[2]Bootstrap learning methods are also called *self-training* or *self-supervised* methods.

Early Work on Bootstrap Learning

One of the earliest approaches to bootstrap learning is by Yarowsky [1995]. This work focuses on the problem of word sense disambiguation. For a given word like “plant,” the task is to label each occurrence of that word with some set of senses (for “plant,” the senses are the biological sense and the manufacturing sense). The system is seeded with a few words that are strongly indicative of a specific sense when collocated with the target word. For example, “life” and “manufacturing” are used as seed words for the two senses of “plant.” All occurrences of “plant” with these two words are labeled, and crucially, all other occurrences of “plant” in documents with labeled occurrences of the word are also assumed to be labeled the same way (this is referred to as the “one sense per discourse” assumption). Then, using all labeled occurrences of “plant’, new indicative words are learned that can be used to label yet more occurrences of the world “plant” based on cooccurrence. This process iterates, labeling more and more unlabeled data.

The Co-Training algorithm uses a bootstrap learning method to classify web pages[3] [Blum and Mitchell, 1998]. Initially, one text classifier is trained over features describing the words on a web page, and another is trained over features describing the words used in hyperlinks pointing to that same web page. In each iteration of the algorithm, the most confident predictions from the classifiers for each view are used to label more documents, and the models are retrained.

The approach used in the Co-Training algorithm is justified by the authors with a claim that it maximizes agreement of the predictions from the two text classifiers over the unlabeled data. Abney [2002] argues that the Co-Training algorithm actually does not directly maximize this agreement, and proposes a different algorithm, the Greedy Agreement Algorithm (GAA), which does. GAA is shown to perform similarly to the Co-Training algorithm and the Yarowsky algorithm on the data set used by Collins and Singer [1999] (which is discussed next).

[3]Note that the Co-Training *setting* does not assume any particular algorithm, but the algorithm used in experiments in the paper was a bootstrap learning algorithm.

Bootstrap Learning for Named Entities and Categories

One of the earliest applications of bootstrap learning in the named entity domain performs named entity classification using parse-based context features [Collins and Singer, 1999]. One of the algorithms presented by Collins and Singer is based on the earlier work of Yarowsky and Blum. It learns rules based on spelling features (e.g., `full-string=California` → `Location`) and contextual features (e.g., `context=partner_at` → `Organization`). Given a small set of initial rules, more and more rules are learned through bootstrap learning, yielding impressive named entity classification accuracies without much initial human effort. However, their approach only covers a limited subset of the noun phrases in a given piece of text, because it only considers noun phrases that occur in certain types of contexts: appositive modifiers (e.g., *Maury Cooper, a vice-president at S&P*), and prepositional phrases (e.g., *a federally funded sewage plant in Georgia*).

Another application of bootstrap learning to the named entity domain is that of Riloff and Jones [1999]. This work describes a method for learning to extract named entity categories from text that has been preprocessed with a shallow parser. Seed instances of each category of interest are provided, which are used to learn new extraction patterns. Those extraction patterns are then used to learn new instances. The process repeats, alternating between learning patterns and learning instances. The idea of *meta-bootstrapping* is also introduced. Meta-bootstrapping runs many iterations of bootstrap learning, then ranks learned patterns or learned words and promotes the top few highest ranked items. The process then repeats. Categories considered in their evaluation are locations, titles (e.g., ceo, cfo, president), and weapons. The results shown are promising, but precision is below 80%.

URL:	`dns.city-net.com/~lmann/awards/hugos/1984.html`
Extraction Template:	“`<i>title</i> by author (`”

Table 2.1: An example extraction URL and template for extracting authors of books learned by the DIPRE system. In the template, “`title`” is the placeholder where the title of a book is extracted, and “`author`” is where the author is extracted.

Bootstrap Learning for Relation Extraction

One of the earliest examples of learning binary relations using bootstrap learning is the Dual Iterative Pattern Relation Expansion (DIPRE) algorithm by Brin [1998]. This work starts with a small set of AuthorOf(author, book) seed instances and bootstraps by discovering extraction templates for web pages. For example, a seed pair is (Isaac Asimov, The Robots of Dawn), and an example URL and extraction template learned are shown in Table 2.1.

Agichtein and Gravano [2000] build on DIPRE to learn the HeadquarteredIn(company, location) relation with their *Snowball* system. *Snowball* improves on DIPRE with more flexible patterns, additional methods of scoring potential patterns and relation instances, and a new evaluation methodology. An experimental evaluation on 300,000 news articles demonstrated 76% precision and recall of 45% for *Snowball* (a confidence threshold could be varied to emphasize precision or recall). Paşca *et al.* [2006b] develop a web-scale approach for extracting the BornIn(person, year) relation from web documents. One of their primary aims was to extract orders of magnitude more relation instances than previous work. Their approach generalizes patterns based on semantic classes of words. For example, their contextual patterns could include snippets like “CL4 written by” where CL4 refers to any one of the set of words {*is, was, has, does, could*}. In their evaluation, 998,992 instances of the target relation are extracted and ranked, with an estimated average precision of 88.38% over that entire set of facts.

Semantic Drift in Bootstrap Learning

After many iterations of running a bootstrap learning algorithm, results often start to exhibit "semantic drift," where errors in labeling during the learning process accumulate and the learned concept drifts from what was initially intended [Curran *et al.*, 2007]. This phenomenon is due to the fact that the learning task is underconstrained: there are many hypotheses which are consistent with the labeled seed data, and many of them are wrong. Coupling the learning of different functions by using positive examples of one function as negative examples for other functions has been shown in some cases to help limit this drift [Riloff and Jones, 1999; Yangarber, 2003]. Such an approach constrains the learning task by providing seed negative examples for each function, and by discovering new negative examples in each iteration of bootstrap learning.

When using bootstrap learning for relation extraction, ensuring that relation arguments are of a certain, expected type can help mitigate the promotion of incorrect instances and thus curb semantic drift. For example, Rosenfeld and Feldman [2007] assume that the argument types for relations of interest are known, and that methods of recognizing those types are available (either rule-based or Conditional Random Field-based sequence taggers). These classifiers are used to validate arguments of potential relation extractions. Rosenfeld and Feldman also use corpus statistics to compare potential arguments to the arguments of seed instances in order to validate them. Paşca *et al.* validate relation arguments using distributional similarity to seed instance arguments.

2.1.2 Semi-Supervised Exponential Models

Logistic regression is an example of an exponential model. Later in the thesis, we discuss methods of exploiting known relations between functions as well as unlabeled data to learn more accurate logistic regression models. Here we review training exponential models on only labeled data, and then review several methods of exploiting unlabeled data to perform semi-supervised learning of exponential models. For a primer on logistic regression, see Mitchell [2006].

Supervised Exponential Models

Exponential models (also called *maximum entropy* when fit using maximum likelihood and *logistic regression* for binary classification problems) are a popular method for discriminative probabilistic modeling. For a binary classifier, we could model the probability of a vector-valued instance x being a positive example of a class y (where y is either 0 or 1) as

$$p_\theta(y = 1|x) = \frac{1}{1 + \exp\left(\sum_k \theta_k x_k\right)}$$

where θ is a parameter vector of the same dimensionality as an instance x.

For multiclass problems, this generalizes to:

$$p_\theta(y_i|x) = \frac{1}{Z(x)} \exp\left(\sum_k \theta_{i,k} x_k\right)$$

where $Z(x) = \sum_i \exp \sum_k \theta_{i,k} x_k$ normalizes the posterior distribution. This formulation is sometimes call *softmax regression*.

Given a collection of labeled training examples L, θ is learned by maximizing the log-likelihood of the labels of the training data:

$$l(\theta, L) = \sum_{(x,y)\in L} \log p_\theta(y|x)$$

Typically the objective function contains the log-likelihood supplemented with a regularization penalty. Here we add a Gaussian prior over the weights (this is commonly referred to as *L2 regularization*:

$$Obj(\theta, L) = \sum_{(x,y)\in L} \log p_\theta(y|x) - \lambda \sum_i \sum_k \theta_{i,k}^2$$

This objective function can be maximized using gradient ascent-based methods, where the gradient with respect to weight $\theta_{i,k}$ is:

$$\frac{\partial Obj(\theta, L)}{\partial \theta_{i,k}} = \sum_{(x,y)\in L} x_i \left[\delta(y=k) - p_\theta(y_k = 1|x)\right] - \lambda\theta_{i,k}$$

Semi-Supervised Exponential Models

Later in the thesis, we will present methods of modifying the supervised objective function discussed above to take advantage of unlabeled data. Several different methods have been proposed by other researchers which fit this general description, and we describe several such methods below. The general idea is to devise a regularization function which encourages some desired behavior over unlabeled examples by penalizing deviations from that behavior, and then add that regularization penalty to the supervised objective function. Optimization can still be performed using gradient-based methods, but the objective function is often no longer convex, so tools like annealing and random restarts are used.

Entropy Regularization

The first example of a semi-supervised exponential model that we discuss here is the work of Smith and Eisner [2007] on feature-based parsing. Their approach uses an exponential model with a wide variety of features of a syntactic parse tree. They find parameters by maximizing an objective function that sums the likelihood of labeled parse trees and a penalty term which penalizes models based on their lack of confidence in their predictions on unlabeled sentences, a method called Entropy Regularization [Brand, 1999; Grandvalet and Bengio, 2005].[4] The penalty term for an unlabeled sentence is the Rényi entropy of the predictions on that example. The authors connect this approach to traditional bootstrap learning. They point out that the gradient for their

[4]This method is similar to Transductive SVMs [Vapnik, 1998], since predictions made for unlabeled data points that are close to the decision boundary of an SVM are lower-confidence than those made for data points far from the boundary.

choice of entropy is much greater for the most confident unsupervised examples than the most unconfident unsupervised examples, and so the most confident examples have the most influence. Thus, it mimics bootstrap learning in that the most confident unsupervised examples influence the learning process.

Label Regularization

Mann and McCallum [2007] consider a multi-class classification setting where few labeled examples are available, but estimates of the prior probability of each class are known. Parameters are fit by optimizing an objective function which sums the conditional likelihood of the known labels, an L2 regularization penalty, and the KL-divergence of the predicted prior probabilities of the different classses from the known priors. Experimental results show that label regularization performs well even with fewer than 10 examples, and generally outperforms supervised L2 logistic regression, Naive Bayes, Naive Bayes with EM, and logistic regression with entropy regularization.

Labeled Features

Druck *et al.* [2008] focus on a setting where labeled *features* are available, rather than labeled examples. For example, a human annotator can declare that 90% of documents that contain the word *puck* should be labeled with class *Hockey*. An objective function consisting of a Gaussian prior on weights plus a penalty on deviations from these declared priors for examples containing the specified features is optimized. The gradient of this penalty is intuitive: it tends to put weight on features which often co-occur with the labeled features. An empirical study shows that, given equal time for labeling, it is more efficient to label features than individual text documents.

Co-Regularization with Multiple Views

In some learning problems, multiple *views* are available, meaning that there are multiple sets of features available. If each view is descriptive enough so that examples can be accurately classified using only that view, then it is possible to use semi-supervised methods that exploit this fact, such as Co-Training [Blum and Mitchell, 1998] (discussed above).

In such multiple-view settings, Sindhwani *et al.* [2005] propose regularizing models with a function that penalizes disagreement in predictions for unlabeled examples. They specifically propose regularizing least squares regression with a penalty which squares the difference between the predictions on each unlabeled example, but using a similar idea with logistic regression is straightforward.

Constraint-Driven Learning

Chang *et al.* [2007] present a framework for semi-supervised learning when domain knowledge is available. For example, when learning a structured output classifier that segments bibliographic citations into fields like *author*, *title*, and *booktitle*, domain knowledge could be expressed through constraints like "Fields cannot end with stop words."

These constraints are incorporated into learning through an objective function that is the sum of a score function and constraint-based penalty terms:

$$\arg\max_{y} \left[\lambda F(x,y) - \sum_{i=1}^{K} \rho_i d(y, 1_{C_i(x)}) \right]$$

where $F(x,y)$ is a vector-valued *feature function* which maps an input/output pair to some feature space, λ is a weight vector (the target of learning), K is the number of constraints, ρ_i is a weight for constraint i, $1_{C_i(x)}$ is the space of labelings which respect the constraint i, and d is some distance between a labeling y and the space $1_{C_i(x)}$.

Learning proceeds using a bootstrap learning algorithm which searches for the top K unlabeled examples which have the highest score function value. The process is "seeded" using a small labeled set.

Their results show that constraints can indeed improve semi-supervised training by penalizing implausible labelings during bootstrap learning. This thesis argues that this idea can be applied at a large scale to couple the training of hundreds or thousands of models.

Posterior Regularization

Ganchev *et al.* [2009] describe a framework called Posterior Regularization (PR) which allows domain knowledge to be specified in the form of constraints over allowable values of latent variables. For example, in semi-supervised learning of an HMM for POS tagging, a constraint can require that the predicted posterior distribution over tag sequences for a sentence contains at least one noun, in expectation. Constraints must be enforced in expectation to allow efficient algorithms for parameter learning; enforcing the constraint so that every labeling with non-zero probability contained a noun, for example, would be intractable. Parameters are estimated using a modification of EM which projects the posterior distribution estimated in the E step onto a space where the constraints are satisfied. The M step remains the same. While PR is a promising framework, for the work with Coupled Logistic Regression (CLR) presented in Chapter 6, it is no more powerful than the penalty used to enforce mutual exclusion for CLR. This is because PR would require constraints to be defined in expectation over the predicted posteriors for mutually exclusive classes, and the latent variables for those posteriors are not connected in the model. Thus, the constraint would be some function of the predicted posteriors, just like the penalty function used.

2.1.3 Machine Learning Frameworks for Constrained Semi-Supervised Learning

Here we discuss a few frameworks for theoretical analysis of semi-supervised learning. These frameworks shed light on what semi-supervised learning is possible and under what conditions it works.

A PAC-style model for Semi-Supervised Learning

Balcan and Blum [2004] present a PAC-style model for learning when both labeled and unlabeled data are available. They assume that in a learning problem, a concept class C is proposed along with some notion of *compatibility* that the chosen concept should have with the data distribution. Unlabeled data can then be useful because it can allow a learner to eliminate concepts in C that violate the required notion of compatibility.

Let $\chi(h, x)$ be a notion of compatibility which is 1 if a hypothesis h is compatible with an example x. Let the unlabeled error rate $err_{unl}(h)$ be defined as $1 - E_{x \in D}[\chi(h, x)]$ where D is the underlying data distribution. Sample-complexity bounds are presented for the case where the target concept c is in the specified concept class C, and the target concept c is also fully compatible with the data distribution. In this case, the number of unlabeled examples necessary for PAC-learning is logarithmic in $|C|$ and the number of labeled examples necessary is logarithmic in $|C_{D,\chi}(\epsilon)|$, the size of the set of concepts in C with $err_{unl} \leq \epsilon$. Thus, the more discriminative the constraint is, the less labeled data required.

Cross-Task Learning with Hints

Daumé [2008] presents a formal model for learning two functions defined over the same instance space when constraints exist between their outputs. If some notion of "compatibility" exists that can identify compatible outputs between the two functions, then unlabeled data can be used to train the models by requiring that the two functions label unlabeled data in compatible ways.

For example, consider the problem of learning two functions $f_1(x)$ and $f_2(x)$ over the same instance space x. Prior knowledge about the compatibility of two outputs y_1 and y_2 is formalized as a constraint function $\chi(y_1, y_2)$ which has value 1 for all correct label pairs for all x with non-zero probability. The usefulness of this constraint is called its discrimination, and is defined as: $Pr[\chi(f_1(x), f_2(x)) = 1]^{-1}$. Results are presented that show that if the constraint has a discrimination that exceeds some lower bound, then the target functions are PAC-learnable. The lower bound depends quadratically on the size of the output spaces.

2.2 Fact Extraction from the Web

The general problem of information extraction from textual data encompasses a broad range of subproblems. Sarawagi [2008] provides a survey of information extraction techniques. For a survey focusing on previous work on named entity recognition, refer to Nadeau and Sekine [2007].

The work in this thesis focuses on extracting general facts about entities from the web. An extraction is considered correct if most humans would agree that it is true, given their background knowledge about the world and a few moments to consult a search engine if they don't immediately know the answer. This is different from some information extraction research, where the task is to annotate individual documents with assertions where they are stated. Thus, we focus our review of previous work on methods for extracting general facts from a (possibly enormous) collection of documents.

First, we discuss methods for extracting instances of categories: contextual extraction pattern-based methods, methods which try to find lists of items on the web, and methods which use distributional similarity. Then we will talk about relation extraction using contextual patterns, followed by discussion of open relation extraction, which is not targetted at any specific relations. We then

Used By	Pattern
Hearst and Etzioni	*cities such as NPList*
Hearst and Etzioni	*such cities as NPList*
Hearst and Etzioni	*NPList* {,} *or other cities*
Hearst and Etzioni	*NPList* {,} *and other cities*
Hearst and Etzioni	*cities* {,} *including NPList*
Hearst and Etzioni	*cities* {,} *especially NPList*
Etzioni	*NP is a city*
Etzioni	*NP is the city*

Table 2.2: Extraction patterns used by Hearst and Etzioni instantiated for the "city" category. Braces indicate optional commas. *NPList* denotes a list of one or more noun phrases.

survey methods of using facts extracted from the web to perform document-level information extraction. Finally, we discuss methods which use ensembles of web extraction techniques.

2.2.1 High-Precision Patterns for Categories

Our approach to fact extraction is based on using high-precision contextual patterns (e.g., X occurring in the context "is mayor of X" suggests that X is a city). One of the first pattern-based approaches to information extraction is that of Hearst [1992]. This approach acquires hyponymy ("is a") relationships from a text corpus using generic extraction contextual patterns. For example, using patterns like "X , such as Y" to extract candidate IsA(X,Y) pairs from an electronic copy of an encyclopedia yields pairs like IsA(waterfowl, ducks) and IsA(protozoa, paramecium). Six hand-coded patterns are used in experiments and are shown in Table 2.2. A method of learning new patterns is sketched out, but not implemented.

KNOWITALL, a system which built on Hearst's work, is created by Etzioni *et al.* [2005]. KNOWITALL uses eight generic extraction patterns to extract candidate hyponym relations from the web using search queries. The eight patterns are shown in Table 2.2. The system then evaluates candidate facts using an extension of the PMI-IR algorithm [Turney, 2001]. Candidates which occur more often with generic patterns are considered more likely to be true. These counts are obtained using search engine hit count queries.

Etzioni *et al.* also present methods of extending the recall of KNOWITALL while maintaining high precision. The methods include pattern learning, subclass extraction, and list extraction. Pattern learning discovers category-specific patterns which are then used both to extract new candidate instances and also to evaluate candidate instances. The pattern learning methods presented later in this thesis are inspired by this pattern learning algorithm. Subclass extraction uses known category instances in generic patterns. For example, if it is known that "biologist" is a type of scientist, then we can use the pattern "biologists such as X" to extract more scientists. Finally, list extraction uses wrapper induction techniques to discover lists of instances on the web, and then extract new candidate instances of a category from the discovered lists. All of these methods improve the recall of KNOWITALL at equivalent levels of precision, and the results demonstrate that the different techniques complement each other well. None of these methods are bootstrap learning methods, because they do not iterate.

Given a collection of extraction patterns and a large corpus, it is not obvious how best to model the probability of some fact being true given the number of times it is extracted by each pattern. Downey *et al.* [2005] present a formal probabilistic model, called URNS, where each occurrence of an instance extracted by a pattern is represented by a ball in an urn. The key question to answer with the model is "If an instance occurs k times out of n draws from the urn, what is the probability that it is a correct extraction?" If the distributions of the number of balls each correct and incorrect extraction have in the urn are known, then this question can be answered with a reasonably straightforward calculation. If not, then the distributions must be estimated, which is done using EM and assumes Zipfian distributions. Multiple extraction patterns are handled using multiple urns.

A pattern like "cities, such as X" is not a perfect extractor of city instances. Consider as an example the sentence “There are numerous problems in cities, such as pollution, crime, and poverty.” This could lead us to believe that pollution, crime, and poverty are cities. Hovy *et al.* [2009] use doubly-anchored generic patterns to increase precision over Hearst’s techniques. The idea is to use known instances of a category in the pattern to increase precision. For example, the pattern “cities, such as Pittsburgh and X” should have very high precision.

Talukdar *et al.* [2006] presents a method for inducing contextual patterns for extracting instances of a category from a set of seed instances. Their method constructs a finite state automaton which represents sequences of words observed near the seed instances in a text corpus. The authors give the example: when trying to learn to extract gene names, one might observe the text sequences: “increased expression of X in vad mice” “the expression of X mrna was greater” and “expression of the X gene in mouse”. An automaton is built starting from the word “expression” (which occurs frequently with seed instances) and is used to induce the patterns: “expression of X in” “expression of X mrna” and “expression of the X gene”. While the example does not illustrate this, their method allows induction of patterns which do not literally occur in the training data. This particular work is of interest because there has been limited work on modeling sequences of words in the domain of web-scale fact extraction.

2.2.2 Other Methods for Category Expansion

List Extraction using Wrapper Induction

As discussed earlier in this section, Etzioni *et al.* use list extraction methods to extend the recall of KNOWITALL. Several other researchers have tried to discover lists and tables of items and learn extraction templates to extract each item in the list or table, in order to expand some set of items of interest. The general idea is to extend a set of instances of a category by using a small set of instances as a query to a search engine, and assume that if they all occur on the same page, then that page is likely to contain a list or table with other instances of that category. For the pages that are returned, typical methods try to find a “wrapper” that can be used to extract the instances used in the query as well as other instances.

An early example of wrapper induction is the work of Doorenbos *et al.* [1997]. More general methods are presented in the thesis work of Kushmerick [1997]. Cohen *et al.* [2002] extend this work by exploiting the HTML structure of pages in wrappers, rather than just prefix and suffix strings. Most of this previous work is focused on information extraction from semi-structured sources. For example, wrappers can be learned to extract structured records containing the author, title, and price of a book from web pages from a specific web site that sells books.

Later work has focused specifically on extending a set of instances of a category or a relation. Etzioni *et al.* [2005] and Nadeau *et al.* [2006] use the HTML structure of a page to induce wrappers for pages that are returned using search engine queries for known instances of a category. The SEAL system of Wang and Cohen [2007] uses character-based prefixes and suffixes as templates in wrappers that can be used to extract from a variety of markup languages. They show that lists of items can accurately be extended in several different languages. They also show good results for a novel ranking method: they rank candidate instances using a random walk-based technique over a graph that connects seeds, web pages they occur on, and candidates extracted by those web pages.

Set Expansion using Distributional Similarity

The *distributional hypothesis* states that the meaning of a word can be characterized by the words that that word co-occurs with in text [Harris, 1954]. This idea is applied computationally to a 6-million word corpus of news stories by Hindle [1990]. Distributional statistics allow Hindle [1990] to determine that, for example, the words “ruling” and “decision” are similar in meaning. Pantel and Lin [2002] use corpus-based distributional similarity between pairs of words to form clusters of words with similar meanings.

The work that exploits distributional similarity that is most relevant to this thesis is that of Pantel *et al.* [2009]. They present an approach to computing pairwise similarities between 500 million terms which is efficient enough to operate in a few days on a 200-node cluster. The learned similarities are then used in a set expansion task. They also study the effects of different variables on the quality of the resulting expanded sets. They find that: the size of the corpus has a significant positive effect on performance; random selections of seed sets of the same size yield wide swings in performance; 5–20 seeds yield much better performance than 2, but going beyond 20 seeds does not lead to additional extraction of correct instances.

2.2.3 Relation Extraction

In this section, we discuss two families of relation extraction methods. Here a *relation* is a predicate with two arguments (e.g., LocatedIn(Carnegie Mellon, Pittsburgh)). First, we discuss pattern-based methods of extracting relational facts from the web. These are similar to the pattern-based methods for categories discussed above. Second, we discuss *open relation extraction* methods, which aim to extract all relations stated in a collection of documents, rather than a pre-specified set of relations of interest.

Pattern-Based Web Relation Extraction

Much previous work focuses on the task of extracting instances of a relation from the web given some seed instances of that relation. These approaches often rely on contextual patterns that are similar to the contextual patterns for extracting category instances discussed above, except that they have two arguments instead of just one.

An early pattern-based web relation extraction approach is the DIPRE system [Brin, 1998]. This system was discussed in greater detail in Section 2.1.1, along with the later work of Agichtein and Gravano and Paşca *et al.*.

A classification-based approach to relation extraction was taken by Mintz *et al.* [2009]. Starting with relation instances from Freebase [Bollacker *et al.*, 2008], sentences containing each known pair of entities in each relation instance are collected, and then feature vectors are generated to represent each relation instance. For example, given the instance (Dusa McDuff, Mathematician) for the "profession" relation, features are extracted from the collection of sentences which mention "Dusa McDuff" and "Mathematician." Features are summed across all occurrences of the instance. The actual features used are conjunctions of features from a dependency parse of the sentence, named entity labels from an off-the-shelf NER package, and lexical features near each argument in the instance. A classifier is then trained for each relation of interest. Average precisions in the 66%–69% range are shown in their evaluation. Mintz *et al.* [2009] built on earlier work by Bunescu and Mooney [2007], which learned to recognize relations expressed in individual sentences using sentences downloaded from the web that contained pairs of entities known to satisfy a relation.

In Section 2.1.1, we discussed the use of type checking to improve the results of relation extraction.

Normand *et al.* [2009] present a bootstrap learning approach to relation extraction which exploits the known arity of each relation of interest (one-to-many or one-to-one). They demonstrate that using an algorithm which leverages this knowledge improves performance.

Open Relation Extraction

Research towards *Open Information Extraction* (Open IE) aims to extract every relation of interest in text, rather than relations from some predefined set of relations of interest. Shinyama and Sekine [2006] discover relations between pairs of entities by clustering documents that mention similar entities and/or express similar relationships between entities, and then aligning the entities in a relational table. For example, given articles about hurricanes, the goal would be to discover a table where one column is the name of a hurricane and the other column is a place damaged by that hurricane. This approach is expensive because it requires pairwise clustering of documents.

A web-scale approach to Open IE is taken by the TEXTRUNNER system [Banko *et al.*, 2007]. The first implementation of TEXTRUNNER uses a Naive Bayes classifier to classify every pair of noun phrases in an input sentence as to whether or not some interesting relation is being expressed between the pair. Heuristics are used to decide which tokens between the pair express the relationship. The Naive Bayes classifier is computationally efficient, so it was feasible to run on sentences from 9 million web pages. This resulted in millions of tuples like *(Oppenheimer, professor of, theoretical physics)* and *(trade schools, similar to, colleges)*. Banko and Etzioni [2008] present an updated implementation of TEXTRUNNER that uses a method called O-CRF, which replaces the Naive Bayes classifier with a Conditional Random Field model. Experimental results show that, with O-CRF, TEXTRUNNER achieves much higher recall and precision than with Naive Bayes.

Clustering of synonymous noun phrases (e.g., *Mars* and *The Red Planet*) and synonymous relations (e.g., *orbits* and *revolves around*) present in the triples extracted by TEXTRUNNER is tackled by the RESOLVER system [Yates and Etzioni, 2009]. RESOLVER uses a model inspired by URNS to assess the probability that two strings are synonymous.

2.2.4 Document-level IE From Web IE

For reasons discussed earlier, this survey has not covered information extraction methods that annotate individual documents. However, there are a few methods that use corpus-level fact extraction techniques to create resources which are then used to annotate individual documents. These are relevant not only because they use corpus-level fact extraction methods, but also because they suggest how to leverage learned facts to perform document-level information extraction.

Nadeau *et al.* [2006] perform named entity recognition using a two step process. First, a seed list of members of a category is expanded using wrapper induction methods. The seeds are used in a query to a search engine to find pages which are potential lists, and then a wrapper is learned for each page based on the HTML structure of the page. This process is iterated in a bootstrap-learning manner. Second, the expanded list of category instances is used in a dictionary-based named entity recognition algorithm. Several heuristics are used to handle issues like cases where a string matches dictionary entries for multiple categories and cases where only partial matches are found. The full algorithm using all heuristics achieved F-scores in the 73–78 range on the MUC-7 corpus.[5] The same approach has been applied to 100 entity types with high accuracies [Nadeau, 2007].

Whitelaw *et al.* [2008] describe a system which learns to annotate individual web documents with named entity labels. The system is trained using a massive collection of training data generated with web IE methods. It starts with a small seed set of instances of entities and relations, and learn patterns from occurrences of the seeds. These patterns are used to generate more labeled instances in web documents. The set of labeled instances is expanded further by assuming that if a labeled entity occurs elsewhere in a document, then the other occurrences should have the same label. Also, if an entity is mentioned in a document that links to a document in which that entity is labeled, then the label is propagated. This process generates hundreds of millions of labeled occurrences of entities in web documents. These labeled occurrences are used to train a classifier which is then used to annotate web documents. The experimental evaluation showed that the system has overall accuracy of 94%.

2.2.5 Ensembles of Web Extraction Systems

Several methods have been presented which extract facts by combining an ensemble of extraction techniques. Ensembles of machine learning methods can yield very good results, often better than the best single method in the ensemble [Dietterich, 2000].

Cafarella *et al.* [2008c] aggregate the outputs of TEXTRUNNER, WEBTABLES, and a deep-web search system. As discussed before, TEXTRUNNER extracts triples of textual strings that

[5]This was a document-level annotation task, rather than a corpus-level task.

express some relation between a pair of entities. WEBTABLES extracts tuples from HTML `table` elements [Cafarella *et al.*, 2008b,a]. The third method submits automatically fills out forms on websites with search forms, and extracts the tables that are returned [Madhavan *et al.*, 2008]. The key idea is that pooling the output of the different extraction methods is worthwhile because they have different strengths in terms of coverage and depth. However, the authors only describe the different methods and point out the value of combining them. Issues like resolution of synomyms are left for future work.

Pennacchiotti and Pantel [2009] present a framework for mixing fact extraction sources called *Ensemble Semantics*. The framework generalizes several previous methods for fact extraction. Results are presented using two different extraction methods: a re-implementation the textual pattern learner from Paşca *et al.* [2006b], and a distribution similarity-based learner [Pantel *et al.*, 2009]. Extractions from these methods were ranked using a variety of feature sources. Results for the categories *actor*, *athlete*, and *musician* showed significant gains from combining the extraction methods and feature sources.

Information from web text and HTML tables is combined in a graph structure by Talukdar *et al.* [2008]. The information from web text comes from open-domain extraction which yields a collection of labeled sets of instances (e.g., “Billy Joel” is a “musician”; these come from the work of Van Durme and Paşca [2008]). The information from HTML tables comes from data provided by Cafarella *et al.* [2008a], and also provides labeled sets of instances. In the graph, instances are connected to classes with weighted edges. For example, “Billy Joel” might be connected to “musician” with weight 0.82, and to “singer” with weight 0.75. Random walks on this graph are used to decide how associated classes are with instances. Since musicians are densely connected with singers in the graph, the two classes reinforce each other in the random walk. An experimental evaluation compares the random walk method with the results of Van Durme and Paşca [2008] and finds that recall improves significantly at equivalent levels of precision.

2.3 Multitask Learning

Multitask learning considers settings where multiple learning tasks are performed simultaneously. The idea is that relationships between different tasks can be exploited by introducing bias based on those relationships. There have been several successes which have demonstrated that supervised learning of multiple related functions together can yield higher accuracy than learning the functions separately with the same number of training examples [Thrun, 1996; Caruana, 1997]. These two approaches both used neural networks with an output unit for each function being learned, and shared hidden units that can leverage shared structure between the functions. The relationship between tasks here is the assumption of some useful common “hidden features” which are useful to multiple tasks. The hidden layers discover the features during training. Both works showed experimental results where multitask learning yielded higher accuracies than independent learning.

In other work with neural networks, a large network was trained to perform many language processing tasks in a multitask fashion by Collobert and Weston [2008]. Multitask learning was accomplished by sharing weights in the first layer of the architecture. Raw input words are fed to the first layer, which maps them into a real-valued space. Weights for this mapping are shared between tasks. In a manner similar to Thrun and Caruana’s work, then, this architecture uses different tasks to discover representations for words which are useful to multiple tasks. Supervised training of the network was performed for part-of-speech tagging, chunking, semantic role labeling, and several other tasks. Semi-supervised training of a language model was performed on the text from Wikipedia. Their results demonstrated that joint learning yielded significantly better performance than the start-of-the-art in semantic role labeling.

A semi-supervised approach to multitask learning was presented by Liu *et al.* [2008]. Their method assumed that the tasks to be learned should have similar parameters. Thus, a shared prior was used for the parameters of all of the different tasks. This encouraged all tasks to have similar parameter values. Positive results were shown on art image retrieval and landmine detection tasks.

The work of Ando *et al.* [2005] learns good “predictive structures” for multiple problems. The idea is that there is some “structural parameter” θ that parameterizes the hypothesis space of each problem being learned. That is, for the l-th problem, the hypothesis space used for the learned predictor is $\mathcal{H}_{l,\theta}$. The goal is to find an optimal parameter θ for which empirical risk for each problem is minimized. For example, θ might map the initial feature space to a lower-dimensional space, encouraging different pairs of features in the initial feature space to have similar weights. θ can be discovered from the data, so that structure between problems can also be discovered from the data, rather than needing to be specified a priori. Semi-supervised applications of Ando et al.’s method are possible, but require that problems without labeled data can have labels automatically generated somehow. This restricts the possible applications of their semi-supervised method.

This thesis takes inspiration from these demonstrations that coupling the supervised and semi-supervised training of multiple functions can improve the learning of those functions. However, we aim to take advantage of explicitly declared constraints that couple functions with more concrete relationships than “these functions should share features” or “these functions should have similar parameters.”

Chapter 3

Coupled Learning of Textual Extraction Patterns

Abstract

In this chapter, we focus on methods of coupling the semi-supervised learning of information extractors that extract information (e.g., City(X) and AthletePlaysForTeam(X, Y)) from free text using textual extraction patterns (e.g., "mayor of X" and "Y star quarterback X"). We identify three general types of coupling among target functions that can be combined to form a dense network of coupled learning problems. We then present an approach in which the input to the learner is an ontology defining a set of target categories and relations to be learned, a handful of seed examples for each, and a set of constraints that couple the various categories and relations (e.g., Person and Sport are mutually exclusive). We show that given this input and millions of unlabeled documents, a semi-supervised learning procedure can achieve very significant accuracy improvements by coupling the training of textual pattern-based extractors for dozens of categories and relations. Based on results reported here, we hypothesize that even greater accuracy improvements will be possible by forming a larger and more dense network of inter-constrained learning tasks. The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous semi-supervised training of both category and relation textual pattern-based extractors and (2) this work proposes that learning many tasks and coupling them as much as possible leads to higher accuracy semi-supervised learning, and provides web-scale experimental evidence to support that point.

3.1 Introduction

In this chapter, we focus on methods of coupling the learning of information extractors that extract information from free text. Machine learning approaches have been shown to be very useful for such learning tasks, including approaches that learn to extract various categories of entities (e.g., Athlete, City, Hobby, Vehicle) and relations (e.g., CompanyProducesProduct, TeamPlaysSport) from text [Bikel *et al.*, 1999; Zelenko *et al.*, 2003]. However, supervised training of accurate entity and relation extractors is costly, requiring a substantial number of labeled training examples for each type of entity and relation to be extracted. This requirement can lead to a large amount of human effort being necessary for each predicate to be learned.

Because of this cost, many researchers have explored semi-supervised learning methods for information extraction that use only a small number of labeled examples of the predicate to be extracted, along with a large collection of unlabeled text [Brin, 1998; Riloff and Jones, 1999; Agichtein and Gravano, 2000]. For example, one popular class of semi-supervised learning methods involves an iterative ”bootstrap learning” process[1] in which initial positive seed examples of the target category (e.g., examples of cities) are first used to search through unlabeled text to identify contextual patterns in which these examples commonly occur (e.g., “mayor of X”). These contextual patterns are then used to find more examples of the predicate, and these steps are iterated to learn a growing set of example instances and corresponding contextual patterns. While such semi-supervised learning methods are promising, they often exhibit unacceptable accuracy because the limited number of initial labeled examples is insufficient to reliably constrain the learning process [Curran *et al.*, 2007].

The idea explored in this chapter is that we can achieve much higher accuracy in semi-supervised learning of information extractors by coupling the simultaneous training of *many* extractors. The intuition here is that the underconstrained semi-supervised learning task can be made easier by

[1]See Section 2.1.1 for a survey of bootstrap learning work.

adding new constraints that arise from coupling the training of many extractors. We identify three general types of coupling among target functions that can be combined to form a dense network of coupled learning problems.

We present an approach in which the input to the semi-supervised learner is an ontology defining a set of target categories and relations to be learned, a handful of seed examples for each, and a set of constraints that couple the various categories and relations (e.g., Person and Sport are mutually exclusive). We show that given this input and millions of unlabeled documents, a semi-supervised learning procedure can achieve very significant accuracy improvements by coupling the training of extractors for dozens of categories and relations. Based on results reported here, we hypothesize that even greater accuracy improvements will be possible by forming a larger and more dense network of inter-constrained learning tasks.

The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous semi-supervised training of both category and relation extractors and (2) this work proposes that learning many tasks and coupling them as much as possible leads to higher accuracy semi-supervised learning, and provides experimental evidence to support that point. This chapter presents material that was originally presented by Carlson *et al.* [2010b].

3.2 Patterns

Our approach uses textual patterns to perform information extraction from free text. In the literature, textual patterns have also been referred to as contextual patterns or lexicosyntactic patterns. In this chapter, we will simply call them *patterns*.

By patterns, we mean literal strings with wildcards that match *noun phrases*.[2] For example,

[2]Noun phrases are defined as strings of tokens that have certain sequences of part-of-speech tags and, in the case of complex phrases like "Procter and Gamble," pass statistical tests of association. See Section 3.4.2 for more details on how noun phrases are defined.

"is mayor of X" is a pattern with literal string "is mayor of" and placeholder X. When applied to the sentence, "Daley is mayor of Chicago" the pattern will extract the noun phrase "Chicago". Patterns can be useful for information extraction because certain patterns are reliable extractors of instances of specific predicates. For example, the previously mentioned pattern "is mayor of X" reliably extracts cities. Patterns can have multiple wildcards, also. For example, in the pattern "X HQ in Y," X is likely to refer to an organization with headquarters in location Y. Such multi-wildcard patterns can be used for extracting relations.

Hearst [1992] presented one of the first pattern-based information extraction methods. This approach acquires hyponymy ("is a") relationships from a text corpus using patterns like "X , such as Y." Applying this method to an electronic version of an encyclopedia yields pairs like IsA(waterfowl, ducks) and IsA(protozoa, paramecium). Only hand-coded patterns are used; a method of learning patterns is discussed but not implemented.

We choose to work with patterns rather than richer representations of context (e.g., patterns with wildcards, or feature-based representations) because the specificity of patterns allows for high precision. It has been shown that high-precision patterns can be learned well given web-scale text. For example, patterns outperformed linguistically richer methods at web-scale in the work of Pantel *et al.* [2004]. Banko and Brill [2001] was one of the first to show that simple methods (like patterns) can perform very well given very large training corpora.

3.3 Bootstrapped Pattern Learning

In this chapter, we focus on a "bootstrapping" method for semi-supervised learning of textual information extraction patterns. See Section 2.1.1 for a discussion of bootstrapping and related work.

3.3.1 Semantic Drift

Bootstrapping approaches to information extraction can yield impressive results [Brin, 1998; Collins and Singer, 1999; Agichtein and Gravano, 2000]. However, after many iterations, accuracy typically declines because errors in labeling accumulate, a problem that has been called "semantic drift" [Curran *et al.*, 2007].

For example, given the category City, a bootstrapping algorithm might be given seeds Chicago, Pittsburgh, Beijing, and Rome. By gathering statistics from occurrences of these seeds, the algorithm can find contexts in which those seeds frequently occur, like "mayor of X" and "city square of X". These new patterns can be used to discover new instances of the City category, and so on. The problem is that there are many patterns which occur frequently with City instances but are too general. For example, the pattern "live in X" occurs frequently with all of the City seeds, but also occurs with instances of other types of locations, like Europe (a continent) and California (a state). Without extra constraints, bootstrapping algorithms frequently diverge because of these overly general extraction patterns.

Example: Semantic Drift

To provide an additional example of the problem of semantic divergence, with real data, consider learning patterns to extract instances of the category Country. We start from seed examples: Austria, Brasil, Canada, Egypt, France, Germany, Indonesia, Iraq, Mexico, Morocco, Netherlands, New Zealand, South Africa, Togo, and United States.

We find all occurrences of these seeds, and extract all patterns that they occur with. We consider the number of times each of those potential patterns occurs, and select the five patterns which occur with at least two of our seeds, and which occur with seeds the largest fraction of all of their occurrences. This is essentially what the UPL algorithm does, discussed later in this chapter in Section 3.4.3.

This strategy yields the five patterns shown in Table 3.1.

Pattern
war with X
ambassador to X
war in X
occupation of X
invasion of X

Table 3.1: Patterns promoted from Country seeds using an uncoupled pattern learning algorithm.

If we then use those patterns to extract new instances of the Country category, we find many noun phrases that are not correct but occur with several of these new patterns. A few examples of such incorrect instances are shown in Table 3.2.

Instance	Patterns Instance Occurs With
planet Earth	“invasion of X”, “occupation of X”, “war with X”
Freetown	“occupation of X”, “war in X”, “invasion of X”, “ambassador to X”
North Africa	“war in X”, “invasion of X”, “occupation of X”

Table 3.2: Instances promoted for the Country category using the patterns from Table 3.1. All are clearly errors. The patterns learned using uncoupled learning are too general.

3.3.2 Coupled Bootstrapped Training

To reduce errors in underconstrained semi-supervised learning, several methods have been considered for adding additional constraints to the learning task. Coupling the learning of category extractors by using positive examples of one category as negative examples for others has been shown to help limit this decline in accuracy [Riloff and Jones, 1999; Yangarber, 2003]. Type checking relation arguments using available entity recognizers can help avoid incorrect labels [Paşca *et al.*, 2006a; Rosenfeld and Feldman, 2007]. Our work in this chapter builds on two ideas and uses them to couple the simultaneous bootstrapped training of many category and relation extractors in multiple ways. Later in the thesis, we explore coupling multiple extraction techniques to further constrain the learning problem. Next, we discuss general types of coupling that can be used to constrain the bootstrapping process.

General Types of Coupling

We have used three general types of coupling:

1. **Output constraints**: For two functions $f_a : X \rightarrow Y_a$ and $f_b : X \rightarrow Y_b$, if we know some constraint on values y_a and y_b for an input x, we can require f_a and f_b to satisfy this constraint. For example, if f_a and f_b are Boolean-valued functions and $f_a(x) \rightarrow f_b(x)$, we could constrain $f_b(x)$ to have value 1 whenever $f_a(x)$ = 1.

2. **Compositional constraints**: For two functions $f_1 : X_1 \rightarrow Y_1$ and $f_2 : X_1 \times X_2 \rightarrow Y_2$, we may have a constraint on valid y_1 and y_2 pairs for a given x_1 and any x_2. We can require f_1 and f_2 to satisfy this constraint. For example, f_1 could "type check" valid first arguments of f_2, so that $\forall x_1, \forall x_2, f_2(x_1, x_2) \rightarrow f_1(x_1)$.

3. **Multi-view-agreement constraints**: For a function $f : X \rightarrow Y$, if X can be partitioned into two "views" where we write $X = \langle X_1, X_2 \rangle$ and we assume that both X_1 and X_2 can predict Y, then we can learn $f_1 : X_1 \rightarrow Y$ and $f_2 : X_2 \rightarrow Y$ and constrain them to agree. For example, Y could be a set of possible categories for a noun phrase, X_1 could represent the string content of that noun phrase, and X_2 could represent the contexts in which that noun phrase appears in web text (this setting is similar to that of Collins and Singer [1999]).

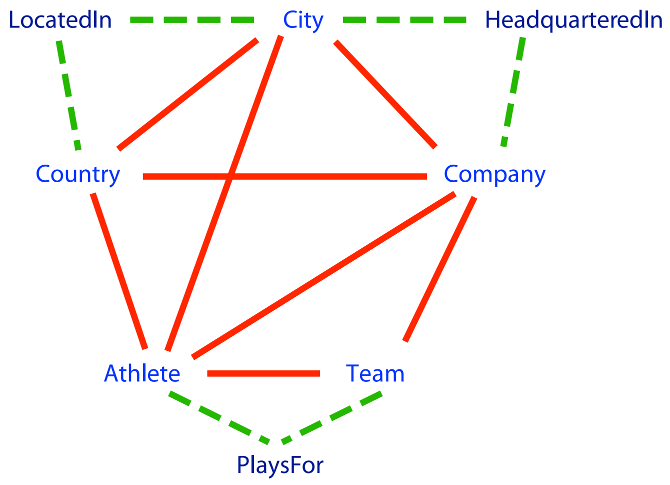


Figure 3.1: Mutual exclusion constraints (solid lines) couple most pairs in the categories Country, City, Company, Team, and Athlete (with City and Team being the exception), while type-checking constraints (dashed lines) couple the relations LocatedIn, HeadquarteredIn, and PlaysFor to the categories specified as their argument types.

Coupling Constraints Used in this Chapter

In this work, the functions that we learn are category and relation extractors, which decide if a noun phrase or pair of noun phrases is an instance of some category or relation (generally referred to as a *predicate* in the rest of this thesis). The general types of coupling discussed above are used to learn these functions in two specific ways:

1. **Mutual Exclusion**: The input to our learner has a list of pairs of predicates which are *mutually exclusive*. These relationships are used to enforce an output constraint over instances: mutually exclusive predicates cannot both be satisfied by the same input x. This is an example of an output constraint.

2. **Relation Argument Type Checking**: We couple the learning of relation extractors with the learning of category extractors using *type checking*. For example, the arguments of the CompanyIsInEconomicSector relation are declared to be of the categories Company and EconomicSector. This is an example of a compositional constraint.

Figure 3.1 illustrates some of the categories, relations, and coupling constraints used in this work.

Example: Coupling Counters Semantic Drift

Continuing the example from above, if we filter out patterns that occur very frequently[3] with noun phrases that are known to be instances of any of the categories that are mutually exclusive with the Country category, we select the five patterns shown in Table 3.3, which are clearly more specific to the Country category than the patterns learned without coupling in Table 3.1. This is essentially the strategy used by CPL, a coupled pattern learning algorithm discussed below.

[3]This is intentionally vague; for the precise criterion used, see the details of CPL in the next section.

Pattern
nations like X
countries other than X
country like X
nations such as X
countries , like X

Table 3.3: Patterns promoted from Country seeds using a coupled pattern learning algorithm. These patterns are clearly more Country-specific than those in Table 3.1, which were learned using uncoupled pattern learning.

3.4 Algorithms

In this section, we present two algorithms with which we investigate the feasibility of improving semi-supervised learning for information extraction with coupling. The general problem addressed by these algorithms is to learn extractors to automatically populate the predicates of a specified ontology with high-confidence instances, starting from a small set of seed instances for each predicate and a large corpus of web pages. We focus on extracting facts that are stated multiple times, which we can assess probabilistically using corpus statistics. We do not resolve strings to real-world entities: the problems of synonym resolution and disambiguation of strings that can refer to multiple entities are left for future work. For example, our algorithms might learn that “Carnegie Mellon University” and “Carnegie Mellon” are both strings that refer to universities, but the algorithms do not learn that the two strings refer to the same entity. We focus our consideration of predicates on unary relations (categories) and binary relations (ones with two arguments, referred to as *relations* in this paper).

3.4.1 Algorithm Inputs

The specific inputs to our algorithms are a large text corpus, and an initial ontology containing the information described in Section 1.4.1.

Algorithm 1: Coupled Pattern Learner (CPL)

```
Input: An ontology $\mathcal{O}$ (see Section 3.4.1 for a description of the information contained in
       the ontology), and text corpus $C$
Output: Proposed instances/contextual patterns for each predicate
for $i = 1, 2, \ldots, \infty$ do
  foreach predicate $p \in \mathcal{O}$ do
    EXTRACT new candidate instances/contextual patterns using recently promoted
    patterns/instances;
    FILTER candidates that violate coupling;
    RANK candidate instances/contextual patterns;
    PROMOTE top candidate instances/contextual patterns;
  end
end
```

3.4.2 Coupled Pattern Learner

The Coupled Pattern Learner (CPL) algorithm learns to extract category and relation instances from unstructured text, and is summarized in Algorithm 1. CPL learns contextual patterns that are high-precision extractors for each predicate (e.g., "X and other software firms" and "X scored a goal for Y") and uses them to grow a set of high-precision predicate instances. Noun phrases that fill in the "X" and "Y" blanks of patterns in sentences in the text corpus are said to *co-occur* with those patterns.

At the start of execution, CPL initializes sets of *promoted* instances and patterns with the seed instances and patterns provided as input. In each iteration, CPL expands these sets of promoted instances and patterns for each predicate while obeying mutual exclusion and type checking constraints. This is accomplished by filtering out candidates that co-occur with instances or patterns from mutually exclusive classes and by requiring arguments of candidate relations to be candidates for the relevant categories. Each step of CPL is discussed in more detail below:

Extracting Candidates

To start each iteration, CPL finds new candidate instances by using the patterns promoted in the last iteration to extract noun phrases that co-occur with those patterns in the text corpus (in the first iteration, the seed patterns are used). To keep the size of this set manageable, for each predicate, CPL selects the 1000 candidates that occur with the most patterns, ignoring instances that have already been promoted. An analogous procedure is used to extract candidate patterns using recently promoted instances.

We use part-of-speech-tag heuristics to limit extraction to instances that appear to be noun phrases and patterns that are likely to be informative. These are described next:

- **Category Instances**: In the blank of a category pattern, CPL looks for a noun phrase by using part-of-speech tags. Common noun phrases are sequences of adjectives (words tagged `JJ`) and common nouns (tagged `NN` or `NNS`), which end in a noun. Proper noun phrases contain proper nouns (tagged `NNP` or `NNPS`) and are allowed to start with a list of common prefixes (e.g., "St." and "Mr.") and end with a list of common suffixes (e.g., ", Inc."). Proper noun phrases that contain prepositions or conjunctions are segmented using a reimplementation of the Lex algorithm [Downey *et al.*, 2007]. Noun phrases that contain only stop words are ignored. Category instances are required to obey the proper/common noun specification of the category. To handle proper nouns like "iPod," if a category has a "proper noun" instance type, instances of that category must contain at least one capital letter. Similarly, if a category has a "common noun" instance type, instances of that category cannot contain any capital letters. [4]

[4]The lists of common prefixes, common suffixes, stop words, and proper nouns containing prepositions or conjunctions are available online.

- **Category Patterns**: When a promoted category instance is found, CPL extracts the preceding words as a candidate pattern if they are verbs followed by a sequence of adjectives, prepositions, or determiners and optionally preceded by nouns (e.g., "being acquired by X" or "companies acquired by X") or nouns and adjectives followed by a sequence of adjectives, prepositions, or determiners (e.g., "former CEO of X"). CPL extracts the words following the instance as a candidate pattern if they are verbs followed optionally by a noun phrase (e.g., "X broke the home run record"), or verbs followed by a preposition (e.g., "X drove to").

- **Relation Instances**: If a promoted relation pattern (e.g., "X is mayor of Y") is found, a candidate relation instance is extracted if both placeholders are valid noun phrases (according to our part-of-speech-tag heuristics), and if they obey the proper/common specifications for their categories.

- **Relation Patterns**: If both arguments from a promoted relation instance are found in a sentence then the intervening sequence of words is extracted as a candidate relation pattern if it contains no more than five tokens, contains at least one word that is not a stop word, and contains an uncapitalized word.

Filtering Candidates using Coupling

Candidate instances and patterns are filtered to enforce mutual exclusion and type checking constraints. A candidate instance is rejected unless the number of times it co-occurs with a promoted pattern is at least τ times more than the number of times it co-occurs with patterns from mutually exclusive predicates, where τ is a parameter that controls how soft the filtering is.[5] This soft con-

[5]In this chapter and the next, we used τ=3.0, but did not perform any sensitivity analysis to arrive at this value. In the work described in Chapter 5, we used $\tau = 10.0$ for more aggressive filtering, but never performed a formal study to find a best value.

straint is much more tolerant of the inevitable noise in web text as well as ambiguous noun phrases than a hard constraint. Candidate patterns for predicates are filtered in the same manner using promoted instances, except that negative instances for a predicate (optionally provided as input to the system) are added to the instances of mutually exclusive predicates used in filtering.

Ranking Candidates

Next, for each predicate CPL ranks candidate instances using the number of promoted patterns that they co-occur with so that candidates that occur with more patterns are ranked higher. Candidate patterns are ranked using an estimate of the precision of each pattern p:

$$Precision(p) = \frac{\sum_{i \in \mathcal{I}} count(i, p)}{count(p)}$$

where $\mathcal{I}$ is the set of promoted instances for the predicate under consideration, $count(i, p)$ is the number of times instance i co-occurs with pattern p in the text corpus, and $count(p)$ is the number of times pattern p occurs in the corpus.

Promoting Candidates

For each predicate, CPL then promotes at most 100 instances and 5 patterns according to the rankings from the previous step. Instances and patterns are only promoted if they co-occur with at least two promoted patterns or instances, respectively. Relation instances are only promoted if their arguments are candidates for the specified categories (that is, they co-occur with at least one promoted pattern for the category, and are not promoted instances of a mutually exclusive category).

Large-Scale Implementation

CPL was designed to allow efficient learning of many predicates simultaneously from a large corpus of sentences extracted from web text. Gathering the statistics needed from the text corpus is the most expensive part of the algorithm. The statistics needed come from two types of queries. First, in the extraction step, CPL has a list of promoted instances and patterns, and needs to know which patterns and instances co-occur with those instances and patterns. Second, in the filtering and ranking steps, CPL needs to know which candidate patterns occur with which promoted instances, and which candidate instances occur with which promoted patterns. CPL gathers these statistics from a preprocessed text corpus which specifies how many times each noun phrase occurs with each category pattern in the corpus, and also how many times each pair of noun phrases occurs with each relation pattern. The preprocessing can be done quickly using using the MapReduce framework [Dean and Ghemawat, 2008]. In each iteration of CPL, CPL gathers corpus statistics from this dataset by scanning through the preprocessed data in two passes: one for extracting candidates and one for counting co-occurrences. CPL can perform one pass in about 15 minutes from a data set derived from 200 million web pages (see Section 3.5.1 for details on the corpus).

3.4.3 Uncoupled Pattern Learner

In our experiments, we use a variant of CPL called Uncoupled Pattern Learner (UPL) which removes the coupling constraints from CPL. Candidates are not filtered using mutual exclusion with other predicates, and relation arguments are not type checked. UPL is equivalent to independent semi-supervised learning of each extractor. The common/proper noun specifications of arguments *are* used to filter out implausible instances.

3.5 Experimental Evaluation

We designed experiments to explore the question: Does coupling learning using mutual-exclusion and type-checking constraints (CPL) improve the performance relative to uncoupled, independent learning (UPL)?

To answer this question, we ran CPL and UPL for 10 iterations of learning. We then compared performance between the pair of methods to see the effects of coupling.

Direct comparison to previous work is difficult for a number of reasons, including the lack of availability of implementations and the lack of a large shared web corpus. However, our evaluation directly tests the usefulness of the coupled approach that we are advocating in this thesis. We believe that the uncoupled baseline is a reasonable and competitive large-scale uncoupled approach to bootstrapped contextual pattern learning.

3.5.1 Experimental Methodology

Input Ontology

The ontology used in all experiments contained categories and relations from two main domains: companies and sports. Extra categories were added to provide negative evidence to the domain-related categories (e.g., Hobby for EconomicSector; Actor, Politician, and Scientist for Athlete and Coach; and BoardGame for Sport) and also to provide wider variety for experiments (e.g., Shape, Emotion). Table 3.6 lists all of the categories in the leftmost column, and Table 3.7 lists the relations in the leftmost column. Categories were initialized with 15 seed instances and 5 seed patterns. The seed instances were specified by the author, and the seed patterns for each category were derived from the generic patterns of Hearst [1992]. Relations were initialized with 15 seed instances, 5 seed negative instances (typically incorrect variations of positive seed examples), and no seed patterns (since it is not obvious how to generate good seed patterns from relation names). Table 3.4 shows the seed instances provided for a few different predicates. Most predicates were declared as mutually exclusive with one another (examples of exceptions include SportsTeam and University; KitchenItem and ProductType; and Company and Product).

Predicate	Seed Instances
city	Antwerp Baghdad Boston Brussels Chicago Frankfurt Hamburg Havana Houston Indianapolis London Moline Moscow Patras Pittsburgh
clothing	dresses shirts pants pant suits skirts gowns hats “ear muffs” socks mittens dress shirt shoes underwear belts
food	tomatoes steak “ice cream” “pumpkin pie” artichokes “cheddar cheese” spinach cookies apples lettuce “bell peppers” popcorn pancakes yogurt celery
stadiumLocatedInCity	(PNC Park, Pittsburgh) (Wrigley Field, Chicago) (Yankee Stadium, New York City) (Wembley Stadium, London) (TD Banknorth Garden, Boston) (Minute Maid Park, Houston) (Bradley Center, Milwaukee) (Wachovia Center, Philadelphia) (Air Canada Centre, Toronto) (Progressive Field, Cleveland) (Coors Field, Denver) (Lambeau Field, Green Bay) (Qualcomm Stadium, San Diego) (Reliant Stadium, Houston) (American Airlines Center, Dallas)
competesWith	(AMD, Intel) (Blockbuster, Netflix) (Citigroup, Morgan Stanley) (Ford, Chrysler) (Ford, Toyota) (Google, Microsoft) (Hershey, Nestle) (Home Depot, Lowes) (Honda, Toyota) (HP, Apple) (Microsoft, Yahoo) (Nestle, Unilever) (Nikon, Canon) (Pfizer, Roche) (Yahoo, Google)

Table 3.4: Seeds used in the CPL evaluation for a few predicates. The full seed lists are available online (see Supplementary Online Materials later in this section).

```
The/DT people/NNS of/IN Boston/NNP reelected/VBD James/NNP
Michael/NNP Curley/NNP when/WRB he/PRP was/VBD in/IN jail/NN ./.

They/PRP were/VBD also/RB forbidden/VBN from/IN harvesting/VBG
right/NN to/TO the/DT very/JJ edge/NN of/IN their/PRP$ fields/NNS
./.

Barrels/NNPS to/TO the/DT back/NN may/MD stiffen/VB
frightened/VBN spines/NNS ./.

I/PRP use/VBP last/JJ ./. fm/JJ integrated/VBN with/IN Amarok/NNP
for/IN my/PRP$ music/NN recommendations/NNS ,/, and/CC I/PRP
highly/RB recommend/VBP it/PRP ./.
```

Table 3.5: Randomly chosen part-of-speech-tagged sentences extracted from the 200-million web page corpus for CPL.

Corpus for CPL

The text corpus used by CPL was from a 200-million-page web crawl.[6] We parsed the HTML, filtered out non-English pages using a stop-word-ratio threshold, then filtered out web spam and adult content using a black list of words. The pages were then segmented into sentences, tokenized, and tagged with parts-of-speech using the OpenNLP[7] package. The sentences were de-duplicated at this point.[8] Finally, we filtered the sentences to eliminate those that were likely to be noisy and not useful for learning (e.g., sentences without a verb, without any lowercase words, with too many words that were all capital letters). This yielded a corpus of roughly 514 million sentences.

As discussed in Section 3.4.2, we processed these sentences to create a data set of noun phrase and contextual pattern co-occurrence counts. To manage the size of the data set, we filtered out

[6]This corpus was an unreleased preliminary version of the 500-million-page ClueWeb09 data set [Callan and Hoy, 2009].

[7]`http://opennlp.sourceforge.net/`

[8]Otherwise, sentences that are parts of templates used to generate pages on a web site can have very large counts and skew the results.

all noun phrases and contexts that only occurred once in the corpus. This yielded a data set that contained 14.9 million unique contextual patterns for categories, 24.6 million unique noun phrases, 232.0 million unique pairs of noun phrases that co-occur together, and 35.7 million unique contextual patterns for relations.

Experimental Procedure

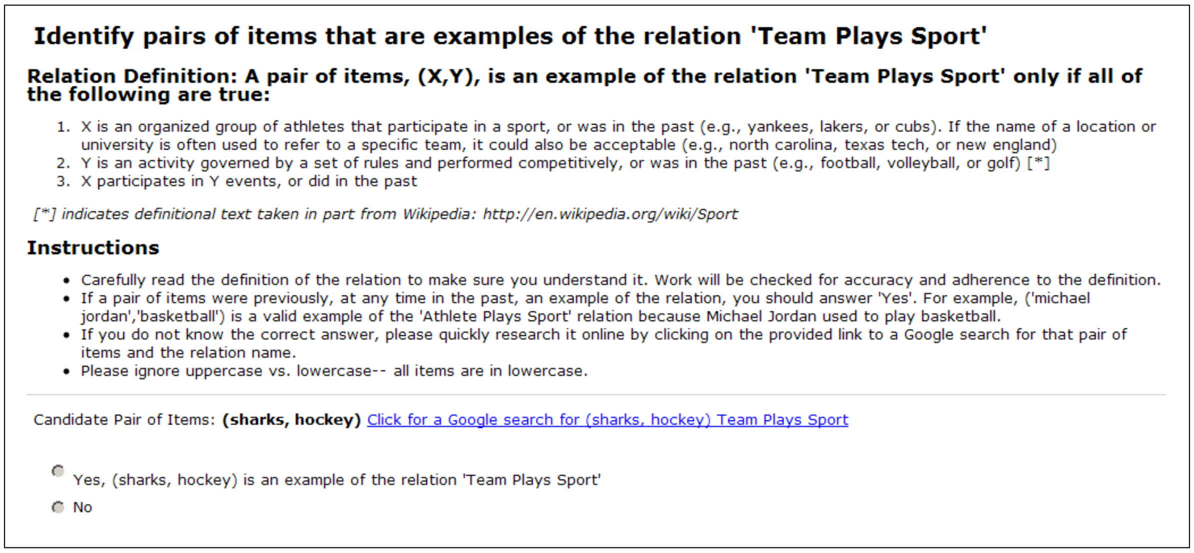
To explore the effects of coupling predicates using mutual-exclusion and type-checking constraints, we compared coupled and uncoupled methods for learning contextual patterns for freeform text: CPL and UPL.

When comparing CPL and UPL, we ran each algorithm for 10 iterations of bootstrapping, and then assessed the instances promoted by the algorithms.

Comparing "All Promotions" for a Predicate To evaluate the precision of all instances promoted by an algorithm on a per-predicate basis, we sampled 30 instances from the set of promoted instances for each predicate, pooled together the samples, and submitted the instances to Mechanical Turk for labeling. This gave an estimate of how accurate all of the instances were and measured the degree to which a particular method avoided "semantic drift". We refer to this method of comparison as the *All Promotions* method in the results below.

Comparing Predicates at "Minimum Recall" We also compared algorithms at matching levels of recall. For each predicate, we only considered the first k instances promoted by each algorithm, where k was the minimum number of instances promoted for that predicate between the two algorithms. For each algorithm and predicate, we sampled 30 instances from the first k promotions, and also submitted them to Mechanical Turk. We refer to this method of comparison as the *Minimum Recall* method in the results below.

While samples of 30 instances do not produce tight confidence intervals for individual estimates



Identify pairs of items that are examples of the relation 'Team Plays Sport'

Relation Definition: A pair of items, (X,Y), is an example of the relation 'Team Plays Sport' only if all of the following are true:

1. X is an organized group of athletes that participate in a sport, or was in the past (e.g., yankees, lakers, or cubs). If the name of a location or university is often used to refer to a specific team, it could also be acceptable (e.g., north carolina, texas tech, or new england)
2. Y is an activity governed by a set of rules and performed competitively, or was in the past (e.g., football, volleyball, or golf) [*]
3. X participates in Y events, or did in the past

[*] indicates definitional text taken in part from Wikipedia: http://en.wikipedia.org/wiki/Sport

Instructions

- Carefully read the definition of the relation to make sure you understand it. Work will be checked for accuracy and adherence to the definition.
- If a pair of items were previously, at any time in the past, an example of the relation, you should answer 'Yes'. For example, ('michael jordan','basketball') is a valid example of the 'Athlete Plays Sport' relation because Michael Jordan used to play basketball.
- If you do not know the correct answer, please quickly research it online by clicking on the provided link to a Google search for that pair of items and the relation name.
- Please ignore uppercase vs. lowercase-- all items are in lowercase.

Candidate Pair of Items: **(sharks, hockey)** Click for a Google search for (sharks, hockey) Team Plays Sport

○ Yes, (sharks, hockey) is an example of the relation 'Team Plays Sport'
○ No

Figure 3.2: A screenshot of the interface used in gathering labels with Mechanical Turk. Each task requested yes/no labels for 15 potential instances of a predicate.

of precision for a single predicate, they should be sufficient for testing for the aggregate effects in which we are interested.

While CPL does not ignore case, our evaluation ignored case, and presented all instances to the evaluators in lower case.

Mechanical Turk Labeling

The various estimates of precision required for our evaluation in this chapter and the next yielded 10717 unique instances. We submitted each of these instances to Mechanical Turk for labeling and had three different individuals label each instance. Mechanical Turk has been shown to be an inexpensive and fast method for obtaining labels for language tasks [Snow *et al.*, 2008]. A screenshot of the UI presented to labelers is shown in Figure 3.2. The majority vote was used to decide the correct label. To estimate the accuracy of the labels produced by this procedure, we sampled 100 instances at random, and manually judged the accuracy of their labels. We found that 96 out of the 100 were correctly labeled using the majority vote. The four errors were: a false positive with "entomology there" labeled as an AcademicField (the labelers ignored the segmentation error), and three false negatives: "informs" as a ProfessionalOrganization, "love seats" as Furniture, and the relation instance "CompanyCompetesWithCompany(bhp, rio)". This suggests that the labels may be biased towards false negatives, which in turn suggests that our precision estimates in the remainder of the paper may be pessimistic.

Results

Table 3.6 gives estimates of the precision of promoted instances for each category for CPL, UPL at full recall (UPL_{full}), and UPL at recall matching CPL ($\text{UPL}_{\text{match}}$), as well as the number of promoted instances for each category after 10 iterations. The "Average" row averages across all predicates for which instances were promoted. The "Weighted Average" is an estimate of the instance-level precision across all predicates obtained by weighting the precision for each predicate by the number of instances promoted for that predicate.[9] Table 3.7 gives this information for each relation, as well. Across all categories and relations, CPL has higher average precision than UPL_{full} and $\text{UPL}_{\text{match}}$. These results suggest that coupling using type checking and mutual exclusion reduces the error rates of the learned extractors.

Another method of comparing which algorithms perform the best is to use the sign test, which is a non-parametric hypothesis test. The test statistic needed to compare CPL with UPL is obtained by counting the number of predicates for which CPL performed better than UPL, and vice versa, ignoring ties. This test gracefully handles predicates where only one method promoted instances: we prefer the method which extracted some instances rather than none for such predicates.

In comparing CPL with UPL for the precision of all promoted instances for each predicate,

[9]The "Average" would be called the macro-average precision, while the "Weighted Average" is an estimate of the micro-average precision.

	Estimated Precision (%)			Promoted Instances (#)	
Predicate	CPL	$\text{UPL}_{\text{match}}$	UPL_{full}	CPL	UPL
AcademicField	70	93	83	46	903
Actor	100	83	33	199	1000
Animal	80	43	50	741	1000
Athlete	87	47	17	132	930
AwardTrophyTournament	57	13	7	86	902
BoardGame	80	80	13	10	907
BodyPart	77	47	17	176	922
Building	33	30	50	597	1000
Celebrity	100	100	90	347	1000
CEO	33	100	30	3	902
City	97	100	100	1000	1000
Clothing	97	87	20	83	973
Coach	93	77	63	188	838
Company	97	80	83	1000	1000
Conference	93	70	53	95	990
Country	57	30	33	1000	1000
EconomicSector	60	40	23	1000	1000
Emotion	77	50	53	483	992
Food	90	53	70	811	1000
Furniture	100	83	0	55	963
Hobby	77	57	33	357	936
KitchenItem	73	27	3	11	900
Mammal	83	80	50	224	1000
Movie	97	70	57	718	1000
NewspaperCompany	90	80	60	179	1000
Politician	80	87	60	178	990
Product	90	67	83	1000	1000
ProductType	73	57	63	712	1000
Profession	73	53	53	916	973
ProfessionalOrganization	93	83	63	104	943
Reptile	95	58	3	19	912
Room	64	48	0	25	913
Scientist	97	97	30	83	971
Shape	77	57	7	43	985
Sport	77	47	13	283	1000
SportsEquipment	20	10	10	58	902
SportsLeague	100	27	7	11	901
SportsTeam	90	53	30	301	903
Stadium	93	90	57	102	767
StateOrProvince	77	60	63	202	1000
Tool	40	13	13	561	1000
Trait	53	63	40	234	1000
University	93	100	97	1000	1000
Vehicle	67	40	30	460	1000
Average	78	62	41	360	960
Weighted average	79	61	42		

Table 3.6: Precision (%) (estimated from 30 random instances) and counts of promoted instances for each category using CPL, UPL at the same level of recall as CPL ($\text{UPL}_{\text{match}}$), and UPL with all promotions (UPL_{full}). Since the number of promotions for a predicate was capped at 100 per iteration, 1000 is the maximum number of instances that could have been promoted in the 10 iteration run.

	Estimated Precision (%)			Promoted Instances (#)	
Predicate	CPL	UPL$_{\text{match}}$	UPL$_{\text{full}}$	CPL	UPL$_{\text{full}}$
CompanyAcquiredCompany	97	87	77	93	230
AthletePlaysForTeam	100	100	93	9	269
AthletePlaysInLeague	-	-	78	0	18
AthletePlaysSport	100	73	47	83	258
CEOOfCompany	100	100	100	18	18
CityLocatedInCountry	93	47	57	185	787
CityLocatedInState	100	67	70	76	194
CoachCoachesInLeague	-	-	-	0	0
CoachCoachesTeam	100	100	100	324	668
CompanyIsInEconomicSector	93	97	97	583	889
CompanyCompetesWithCompany	100	75	67	28	123
CompanyHasOfficeInCity	-	-	63	0	526
CompanyHasOfficeInCountry	-	-	90	0	195
CompanyHeadquarteredInCity	50	100	53	2	532
LeaguePlaysGamesInStadium	-	-	-	0	0
CompanyProducesProduct	97	97	93	54	215
ProductInstanceOfProductType	73	77	67	153	484
SportUsesSportsEquipment	33	13	3	15	1330
StadiumLocatedInCity	100	86	20	7	600
StateHasCapitalCity	60	70	70	266	188
StateLocatedInCountry	97	53	40	194	1299
TeamHasHomeStadium	100	100	87	97	208
TeamPlaysAgainstTeam	100	90	80	238	2088
TeamHasHomeCity	-	-	57	0	680
TeamPlaysInLeague	100	86	67	7	255
TeamPlaysSport	-	-	70	0	177
TeamWonAwardTrophyTournament	90	57	70	128	262
Average	89	79	69	95	463
Weighted Average	91	81	61		

Table 3.7: Precision (%) (estimated from 30 random instances) and counts of promoted instances for each relation using CPL, UPL at the same level of recall as CPL (UPL$_{\text{match}}$), and UPL with all promotions (UPL$_{\text{full}}$).

	All Promotions		Minimum Recall	
Comparison	Wins	p-value	Wins	p-value
CPL vs. UPL	55 vs. 12	1.03e-07	46 vs. 10	1.25e-06

Table 3.8: CPL and UPL compared based on the precision of all promotions for each predicate (All Promotions) and the precision of the instances promoted cut off at the minimum recall out of the pair for each predicate (Minimum Recall). Wins record how many predicates had superior precision for each method, and the p-value according to a sign test is given. All results are statistically significant at the 5% level.

Software
- isA: Product Type, Economic Sector
- productInstances: iTunes, Excel, Adobe Photoshop, Microsoft Outlook, AutoCAD, Kazaa
- companiesInSector: Infosys, SAP, Microsoft, IBM, Wipro, Symantec

Tigers
- isA: Mammal, Sports Team
- teamHomeStadium: Comerica Park
- teamCoach: Les Miles
- teamWonTrophy: World Series
- teamPlaysAgainstTeam: Yankees, Royals, Sox, White Sox, Red Sox, Warriors

Figure 3.3: Examples of promoted facts about two entities. The categories (listed under the "isA" slot) of "Software" were given in the seed ontology; all other facts were discovered by CPL. Values shown for "productInstances" and "companiesInSector" for "Software" are a subset of the full set of promoted values.

as well as the "minimum recall" sample discussed above, CPL performs statistically significantly better than UPL for both methods of sampling. Comparing all promotions, 55 predicates have higher precision for CPL, and 12 predicates have higher precision for UPL, giving a p-value of 1.03e-07. In comparing at Minimum Recall, 37 predicates are better for CPL vs. 8 for UPL, giving a p-value of 1.54e-05. These results confirm that coupling yields significantly higher accuracies across all predicates than using independent, uncoupled learning.

Figure 3.3 gives some examples of the type of information extracted in our experiments for two noun phrases, "Software" and "Tigers." The initial seed examples provided specified that "software" is a ProductType and an EconomicSector; the rest of the information in the figure was extracted by CPL. To provide more examples of facts learned by CPL, Table 3.9 shows facts promoted during the run of CPL, selected uniformly at random, along with the iteration in which

Predicate	Iteration	Instance
country	2	solomon islands
company	2	nbc universal
product	5	stuffit
economicSector	7	marine industry
clothing	9	collar shirt
cityLocatedInState	2	(charlotte, north carolina)
sportUsesEquipment	3	(soccer, player)
companyEconomicSector	3	(unocal, oil)
companyEconomicSector	4	(emc, storage)
productInstanceOf	5	(final cut pro, software)

Table 3.9: Example beliefs promoted by CPL at various iterations. All are correct, except for sportUsesEquipment(soccer,player).

Predicate	Pattern
actor	film version , starring X
athlete	blockbuster trade for X
company	airlines , including X
emotion	personal feelings of X
hobby	hobbies include X
acquired	X announced plans to buy Y
athletePlaysSport	X learned to play Y
ceoOf	Y chairman X
companyEconomicSector	Y giants X
teamPlaysInLeague	X dominance in Y

Table 3.10: Example free-text patterns learned by CPL. X and Y represent placeholders for noun phrases to be extracted.

they were promoted.

Table 3.10 shows patterns learned by CPL during the run for selected categories and relations.

Supplementary Online Materials

Several different types of materials from our evaluation are posted online at `http://rtw.ml.cmu.edu/acarlson_thesis`:

- The input ontology, containing all categories and relations. See Section 1.4.1 for a description of the information specified for each category and relation.
- The lists of common prefixes, common suffixes, stop words, and proper nouns connected by prepositions or conjuctions, used to segment noun phrases.
- All instances promoted by CPL and UPL.
- All textual patterns promoted by pattern learning in the CPL and UPL experiments.
- Browseable knowledge bases in XML format of all promoted instances and candidate instances from the runs of CPL, with patterns and URLs that extracted each instance.
- All judgments obtained from Mechanical Turk.
- An example screenshot from a Mechanical Turk task.
- Templates used to create the Mechanical Turk tasks, which may be of general use.

3.5.2 Discussion

Error Analysis

The categories for which CPL has the most difficulty (e.g., ProductType, SportsEquipment, Traits, Vehicles) tend to be common nouns. We expect that a more complete hierarchy of common nouns would better constrain these categories and yield better accuracies.

CPL generally had high accuracies for relations, but suffered from sparsity. SportUsesSports-Equipment suffered because the SportsEquipment category performed poorly, resulting in bad type checking. StateHasCapital and CompanyHeadquarteredInCity drifted to the more general relations of StateContainsCity and CompanyHasOperationsInCity. These latter two cases can be improved by adding the ability to infer negative examples using the knowledge that these are functional relations: patterns that extract multiple capitals for the same city could be filtered out using this knowledge.

Our experiments included five relations for which no instances were promoted by any algorithms: CoachCoachesAthlete, AthletePlaysInStadium, CoachWonAwardTrophyTournament, SportPlaysGamesInStadium, and AthleteIsTeammateOfAthlete. These relations show that some relations are not easy to extract using the extraction methods used in this paper. However, many of these relations could be inferred from instances promoted for other relations. We investigate learning to infer such relations later in the thesis, in Chapter 5.

CPL as a Case Study of Coupled Semi-Supervised Learning

The results presented above demonstrate that coupling improves the precision of CPL's learned extractors. A key question for discussion is: *How does our work with CPL relate to general theory on semi-supervised learning?*

As discussed in Section 2.1.3, theory tells us that if we know that the target functions that we are trying to learn must be *compatible* with some constraints, we can use unlabeled data to rule out hypotheses that correctly label the labeled data but are incompatible with the constraints over unlabeled data. By using unlabeled data in this way, we can reduce the amount of labeled data we need to learn our target functions in the PAC sense. These general theoretical ideas are supported by both Balcan and Blum [2004] and Daumé [2008].

Our answer to the question posed above, then, is that our results with CPL serve as a case study of coupled semi-supervised learning of dozens of classifiers, and are supported by theory on semi-supervised learning. Our argument is presented next:

- **Correctness of compatibility assumptions:** In our case study with CPL, it is reasonable to assume that accurate classifiers will be generally compatible with our coupling constraints. This isn't always true, particularly since we do not disambiguate between homonyms (entities with same strings, e.g., "China" the country and "China" the dishes that people eat off of). CPL uses mutual exclusion constraints in a fuzzy way when learning patterns to compensate.
- **CPL rules out hypotheses incompatible with coupling constraints:** CPL learns classifiers that label noun phrases based on the contextual patterns that they co-occur with, and based on their string content (these are rote classifiers that label a noun phrase like "Carnegie Mellon" based solely on its identity). The bootstrapping process expands the set of patterns used in the pattern-based classifiers, as well as the set of noun phrases that are labeled using the rote learners. This process gradually labels the unlabeled data in ways consistent with the labeled data and with the mutual exclusion and type-checking coupling constraints. By filtering out instances and patterns that violate mutual exclusion and type checking constraints, CPL does indeed rule out hypotheses that are incompatible with the coupling constraints.

Thus, we argue that our results with CPL serve as a case study of coupled semi-supervised learning of dozens of classifiers, that our case study demonstrates that coupling can improve the accuracy of the learned classifiers, and that our results agree with theoretical results on semi-supervised learning.

3.6 Conclusion

We have presented methods of coupling the semi-supervised learning of category and relation instance extraction patterns and demonstrated empirically that coupling forestalls the problem of semantic drift associated with bootstrap learning methods. This empirical evidence leads us to advocate large-scale coupled training as a strategy to significantly improve accuracy in semi-supervised learning.

Chapter 4

Coupling Wrapper Induction and Multiple Extractors

Abstract

In this chapter, we first consider applying the ideas presented in the previous chapter for coupling pattern-based information extraction to a different type of extraction method, wrapper induction for semi-structured web pages. We explore using coupling constraints based on mutual exclusion and type checking of relation arguments to learn more accurate wrappers within a bootstrap learning process. We then consider how to couple multiple extraction methods that typically make independent errors: the method of pattern-based extraction from unstructured text performed by CPL and the method for wrapper-based extraction from semi-structured documents which we discuss first in this chapter. To couple these two methods, we use a strategy that only promotes instances extracted by both methods. Experimental results on dozens of categories and relations demonstrate that coupling wrapper induction improves the precision of the promoted facts, and that coupling multiple extraction methods leads to higher precision than either of the methods alone. The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous training of multiple wrapper inducers (2) this work is the first to couple the simultaneous training of multiple extraction methods (pattern-based extraction from free text and wrapper-based extraction from semi-structured documents), rather than simply combining the output of multiple extraction methods after training.

4.1 Introduction

In Chapter 3, we saw that coupled semi-supervised learning could improve extraction pattern learning for many categories and relations. A logical question that follows from this work is: *Can the same ideas of mutual exclusion and type checking be used to create coupling constraints that improve a different extraction technique?*

To answer this question, in this chapter, we first consider applying the ideas presented in the previous chapter for coupling pattern-based information extraction to a different type of extraction method, wrapper induction for semi-structured web pages. Semi-structured web pages are web pages that contain lists or tables of information. Such pages are often generated from structured data records by rendering them using a template. If regularities in the template can be found, they can be exploited to extract the original structured data. The problem of discovering the template and extracting the structured records is called *wrapper induction*, because the template *wraps* the data presented in the page. Section 2.2.2 discusses some of the literature on wrapper induction. We explore using the same types of coupling constraints used in the previous chapter for CPL, mutual exclusion and type checking of relation arguments, to learn more accurate wrappers within a bootstrap learning process.

Another logical question that follows from the work in Chapter 3 is: *Are there other types of coupling constraints that can be exploited to improve semi-supervised learning, beyond mutual exclusion and type checking constraints?*

To answer this question, we explore how to couple multiple extraction methods that typically make independent errors: the method of pattern-based extraction from unstructured text performed by CPL and the method for wrapper-based extraction from semi-structured documents which we discuss first in this chapter. Noun phrases on the web appear in multiple types of contexts, including freeform textual contexts and semi-structured contexts. For example, the noun phrase “Pittsburgh” occurs on the web with a distribution of freeform textual contexts such as “mayor of X”, and it also appears with a distribution of semi-structured contexts such as the HTML tags for a list item at a particular URL. We assume that either of these distributions is sufficient to classify a noun phrase. We therefore employ a method that trains two noun phrase classifiers, one using each type of context distribution, and requires that the two classifiers agree on the label for each given noun phrase. This is an example of a multi-view constraint, as discussed in Section 3.3.2.

We experimentally evaluate these ideas by running three algorithms for 10 iterations using the same methodology that was used to compare CPL and UPL in Chapter 3. First, we run SEAL [Wang and Cohen, 2009], a state-of-the-art wrapper induction system, in an uncoupled bootstrapping loop. Second, we run Coupled SEAL (CSEAL), which adds coupling to SEAL by filtering out wrappers that extract candidates that violate mutual exclusion relationships, and by checking the types of candidate relation instances using their respective categories. Finally, we run Meta-Bootstrap Learner, which runs both CSEAL and CPL, and only promotes candidates suggested by both methods. Results show that coupling SEAL improves its precision, and that coupling CSEAL and CPL results in higher precision than either method alone.

The main research contributions of the chapter are: (1) this work is the first to couple the simultaneous training of multiple wrapper inducers and demonstrate that mutual exclusion and type checking constraints can improve extraction methods other than pattern learning, and (2) this work is the first to couple the simultaneous training of multiple extraction methods (rather than simply combining the output of multiple extraction methods after training, as in Cafarella *et al.* [2008c] and Pennacchiotti and Pantel [2009]). This chapter presents material that was originally presented by Carlson *et al.* [2010b].

Algorithm 2: Set expansion using wrapper induction

```
Input: Seed instance set $\mathcal{S}$, and text corpus $C$
Output: Candidate instances that extend the set

QUERY for documents in $C$ containing seed instances in $\mathcal{S}$;
foreach returned document $d$ do
    FIND occurrences of seeds in $d$;
    if seeds occur in a consistent wrapper then
        EXTRACT new candidates using wrapper and add them to the results;
    else
        DISCARD the document;
    end
end
```

4.2 Coupled Wrapper Induction

In this section, we discuss SEAL [Wang and Cohen, 2009], an off-the-shelf, state-of-the-art method for wrapper induction, and Coupled SEAL, an algorithm that calls SEAL as a subroutine and couples the learning of functions using mutual exclusion and type checking constraints. To be clear, *wrapper induction* refers to the problem of discovering a template and extracting structured records from a document that contains structured records presented in an automatic, template-driven way. A *wrapper* refers to the template used to generate the document, because the template *wraps* the data presented in the page. Once the wrapper is discovered, it can be used to extract the original structured data from a document. To ground the discussion, we start with a discussion of how wrapper induction works and present an example.

4.2.1 A Wrapper Induction Example

Our methods of wrapper induction use the steps shown in Algorithm 2 and illustrated in Figure 4.1. To show how these steps proceed, we present a simple example in the rest of the section. Imagine that we want to find members of the category State, and that we already know that Illinois, California, West Virginia, Hawaii, Alaska, and Florida are instances of the category. The first step is

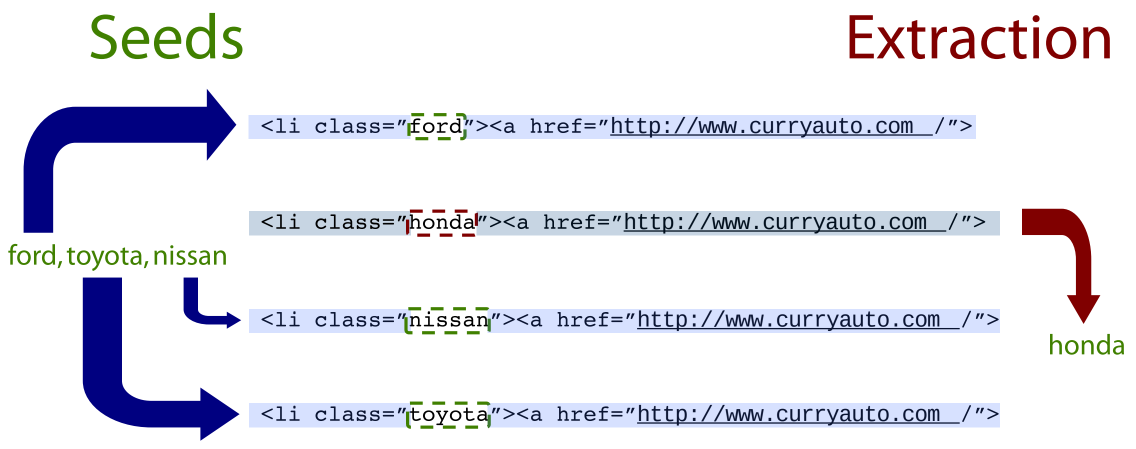

Seeds
Extraction
ford, toyota, nissan
honda
<li class="ford"><a href="http://www.curryauto.com_/">
<li class="honda"><a href="http://www.curryauto.com_/">
<li class="nissan"><a href="http://www.curryauto.com_/">
<li class="toyota"><a href="http://www.curryauto.com_/">

Figure 4.1: An illustration of wrapper induction. Seeds are located in a retrieved document, a template consisting of a prefix and a suffix is found, and new candidate instances are extracted using the template.

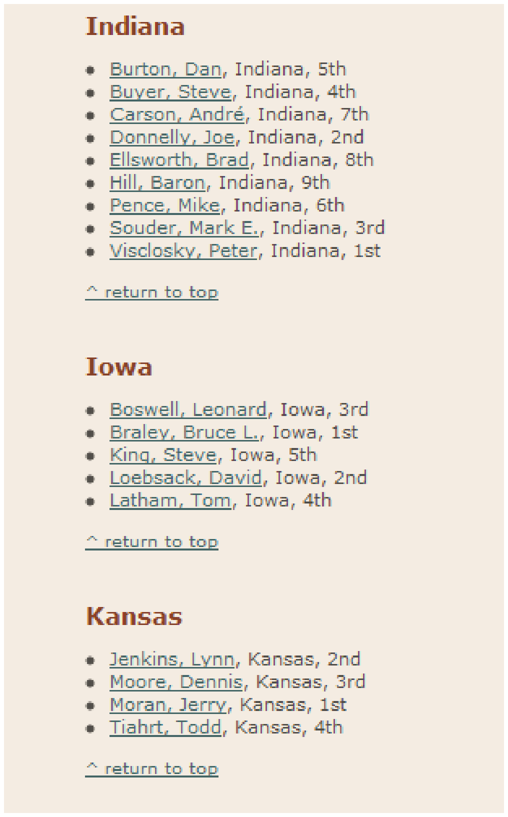
to find web pages that are likely to contain lists of other states. So, our program queries a search engine with the (conjunctive) query "illinois california west virginia hawaii alaska florida". Pages that mention all of these entities are likely to contain a list or table that mentions other instances. The top result is a document with URL: `http://www.house.gov/house/MemberWWW_by_State.shtml`. This page lists members of the United States House of Representatives by state. A screenshot of a portion of the page is shown in Figure 4.2.

Next, our program examines the HTML source of the page. The first seed, Illinois, is found in multiple places in the source code, including these snippets:

- `<H3><A name="il" id="il"></A>Illinois</H3>`
- `Bean, Melissa L.</A>, Illinois, 8th</LI>`
- `Biggert, Judy</A>, Illinois, 13th</LI>`

The second seed, California, is also found in multiple places in the source code, including:

- `<H3><A name="ca" id="ca"></A>California</H3>`
- `Baca, Joe</A>, California, 43rd</LI>`



Indiana

- Burton, Dan, Indiana, 5th
- Buyer, Steve, Indiana, 4th
- Carson, André, Indiana, 7th
- Donnelly, Joe, Indiana, 2nd
- Ellsworth, Brad, Indiana, 8th
- Hill, Baron, Indiana, 9th
- Pence, Mike, Indiana, 6th
- Souder, Mark E., Indiana, 3rd
- Visclosky, Peter, Indiana, 1st

^ return to top

Iowa

- Boswell, Leonard, Iowa, 3rd
- Braley, Bruce L., Iowa, 1st
- King, Steve, Iowa, 5th
- Loebsack, David, Iowa, 2nd
- Latham, Tom, Iowa, 4th

^ return to top

Kansas

- Jenkins, Lynn, Kansas, 2nd
- Moore, Dennis, Kansas, 3rd
- Moran, Jerry, Kansas, 1st
- Tiahrt, Todd, Kansas, 4th

^ return to top

Figure 4.2: An example of semi-structured web page content.

- `Becerra, Xavier</A>, California, 31st</LI>`

The other seeds are found in similar contexts.

Based on these snippets, our program can induce two different wrappers, where `[X]` represents the placeholder used to extract items:

- `></A>[X]</H3>`
- `</A>, [X],`

Applying these wrappers to the page yields all 50 states of the United States. This is a typical example of how wrapper induction is done.

However, Guam, Puerto Rico, American Samoa, and Virgin Islands are also extracted, which are territories of the United States, rather than states. It turns out that this page is too general; it

URL:	`http://www.shopcarparts.com/`
Wrapper:	`.html" CLASS="shopcp">[X] Parts</A> <br>`
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chevy, chrysler, daewoo, daihatsu, dodge, eagle, ford, ...
URL:	`http://www.allautoreviews.com/`
Wrapper:	`</a><br> <a href="auto_reviews/[X]/`
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chrysler, dodge, ford, gmc, honda, hyundai, ...
URL:	`http://www.hertrichs.com/`
Wrapper:	`<li class="franchise [X]"> <h4><a href="#">`
Content:	buick, chevrolet, chrysler, dodge, ford, gmc, isuzu, jeep, lincoln, mazda, mercury, ...

Table 4.1: Examples of wrappers constructed by SEAL for various web pages given the seeds: Ford, Nissan, Toyota. In the table, $[X]$ is a placeholder for extracting instances.

lists states and territories of the United States, rather than just states. If we knew that Guam was not a state, we could use that knowledge to decide that this web page is too general. We use this idea below to couple the learning of wrapper inducers for multiple predicates using mutual exclusion relationships.

It is often the case that seed instances will occur together on a page but not actually be embedded in a semi-structured format. These cases are usually easily detected because there will not be a consistent template which extracts all of the seeds. Wrapper induction methods will typically ignore such pages and not extract anything from them.

4.2.2 SEAL

SEAL [Wang and Cohen, 2009] is a set-expansion system that accepts input elements (seeds) of some target set S and automatically finds other probable elements of S in semi-structured documents such as web pages by querying the web using the seeds. The algorithm implemented in

SEAL constructs page-specific extraction rules, or *wrappers*, that are independent of the human language and markup language of the web pages. It uses methods based on those presented in Algorithm 2. SEAL can expand sets of category instances as well as binary relation instances. Every category wrapper is defined by character strings, which specify the left context and right context necessary for an entity to be extracted from a page. Relation instance wrappers also are defined using an infix context that separates the two arguments of the instance (e.g., a wrapper might be "`<li>`$[X]$ `:` $[Y]$`</li>`" where " `:` " is the infix context which separates the two arguments). These context strings are selected to be maximally-long contexts that bracket at least one occurrence of every seed on a page. Table 4.1 shows a few examples of such wrappers for categories. An instance is extracted by a wrapper if it is found anywhere in the document with left and right context identical to that of the wrapper.

When given large sets of seeds, SEAL can be configured to "subsample" the seeds some number of times [Wang and Cohen, 2008]. Subsampling samples a subset of the seeds and uses that subset as a query to a search engine, which is necessary because using all examples in one query would typically not yield any matched results.

In our work, we used code for SEAL provided by its authors.

4.2.3 Coupled SEAL

SEAL does not have a mechanism for exploiting mutual-exclusion or type-checking constraints. Wrappers for each predicate are learned independently in SEAL. Our algorithm, Coupled SEAL (CSEAL), adds these constraints on top of SEAL. CSEAL is summarized in Algorithm 3. In each iteration of bootstrapping, we invoke SEAL using the recently promoted instances. SEAL returns a list of new candidate instances and documents that they were extracted from. CSEAL filters out any document that extracts a candidate instance that is a member of a mutually exclusive predicate. Additionally, CSEAL only considers candidate relation instances if their arguments are candidate

Algorithm 3: Coupled SEAL (CSEAL)

```
Input: An ontology $\mathcal{O}$, and text corpus $C$
Output: Trusted instances/wrappers for each predicate
for $i = 1, 2, \ldots, \infty$ do
  foreach predicate $p \in \mathcal{O}$ do
    begin Call existing SEAL code to:
      QUERY for documents containing recently promoted instances;
      foreach returned document $d$ do
        LEARN wrapper for $d$;
        if seeds occur in a consistent wrapper then
          EXTRACT new candidates using wrapper;
        else
          DISCARD the document;
        end
      end
    end
    DISCARD wrappers that extract candidates that violate coupling;
    RANK candidate instances;
    PROMOTE top candidates;
  end
end
```

instances for the respective category types, and are not already promoted instances of categories that are mutually exclusive with the types. These forms of coupling should filter out cases where a subsampled set of seeds happens to occur on a page but that page does not in fact contain a valid list of predicate instances. They should also filter out cases where instances of a predicate that is more general than the one being learned are listed (e.g., if a long list of locations of various types is present on a page, but we are learning some specific type of location, such as “city”).

After filtering, CSEAL ranks all candidate instances by the number of unfiltered wrappers that extracted them[1], and promotes at most 100 instances that were extracted by at least two wrappers

[1]SEAL has other scoring and ranking methods available, but we found that they have modes of failure the led to problems in our bootstrapping setting. Specifically, SEAL would often return long lists of common nouns with scores very close to 1.0 for certain categories (e.g., Tools, Sports Equipment.

(ties are broken arbitrarily). To deal with sets of web pages from the same domain that repeat the same list, only one page from a domain is counted in ranking candidates. Without limiting consideration to domains, navigational and other template-generated elements that repeat many times can dramatically skew the results.

In our experiments below, CSEAL refers to the algorithm described here, and SEAL refers to CSEAL without the filtering step: SEAL does not filter out wrappers that extract candidates that violate mutual-exclusion relations, and SEAL does not enforce relation instance type checking.

4.3 Coupling Multiple Extractors

In this section, we present a method for coupling multiple extraction methods that typically make independent errors: the method of pattern-based extraction from unstructured text performed by CPL and the method for wrapper-based extraction from semi-structured documents which we discuss first in this chapter. CPL will often make errors due to improper noun phrase segmentation (e.g., extracting only “Journal” from “publications such as the Journal of Artificial Intelligence”); CSEAL does not tend to make such errors because CSEAL learns templates with prefixes and suffixes. CPL will also make errors due to complex language (e.g., extracting “pollution” as a city from “problems that affect cities, such as pollution” due to the pattern “cities , such as X”). CSEAL will typically make errors when it thinks it has found a consistent context that indicates a list, but actually has not. For example, if CSEAL learns a wrapper that consists of list item (`<li>`) tags, CSEAL will often extract all list items on a page. These different error cases are quite different, which makes it reasonable to assume that CPL and CSEAL make independent errors.

Algorithm 4: Meta-Bootstrap Learner (MBL)

```
Input: An ontology $\mathcal{O}$, a set of extractors $\mathcal{E}$
Output: Trusted instances for each predicate
for $i = 1, 2, \ldots, \infty$ do
    foreach predicate $p \in \mathcal{O}$ do
        foreach extractor $e \in \mathcal{E}$ do
            EXTRACT new candidates for $p$ using $e$ with recently promoted instances;
        end
        FILTER candidates that violate mutual-exclusion or type-checking constraints;
        PROMOTE at most 100 candidates that were extracted by all extractors;
    end
end
```

4.3.1 Meta-Bootstrap Learner

To couple the learning of CPL and CSEAL, we use an algorithm called Meta-Bootstrap Learner (MBL). MBL couples the training of multiple extraction techniques using a multi-view constraint (as discussed in Section 3.3.2) that requires them to agree. Here textual pattern features and semi-structured document features provide two different views.

MBL is summarized in Algorithm 4. In this chapter, the subordinate algorithms used with MBL are CSEAL and CPL. When using CSEAL and CPL with MBL, the subordinate algorithms do not promote instances on their own. Instead, they skip the promotion step and report evidence about each candidate to MBL, and MBL is responsible for promoting instances. MBL uses a simple combination method: MBL promotes any instance that has been recommended by both techniques while obeying the mutual-exclusion and type-checking constraints specified in the ontology. MBL will promote up to 100 instances of a predicate in each iteration. If more than 100 candidates are eligible for promotion, the ones that occur with the largest sets of distinct patterns and wrappers are chosen. The constraints added to the learning problem by MBL are illustrated in Figure 4.3.

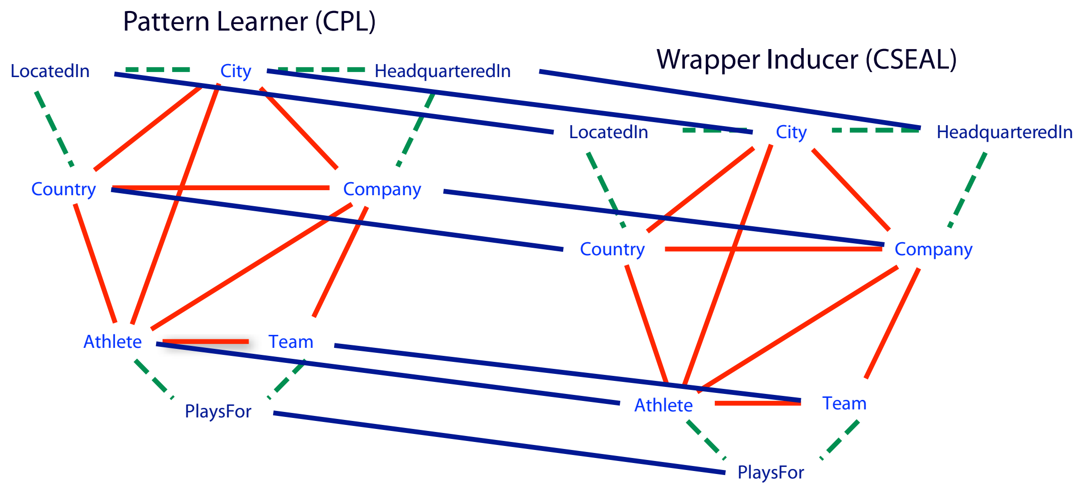


Figure 4.3: CPL and CSEAL each exploit coupling between their learned functions, indicated by lines among them. Since CPL and CSEAL each learn approximations of the same functions, we can use a multi-view constraint between each pair, illustrated by lines connecting each pair in this figure.

4.4 Experimental Evaluation

We devised experiments to explore the following questions:

- Do mutual-exclusion and type-checking constraints improve the performance of CSEAL relative to the uncoupled methods of SEAL?
- Does MBL achieve better performance than CPL and CSEAL by combining their outputs with a multi-view constraint?

To answer these questions, we ran CSEAL, SEAL, and MBL with CPL[2] and CSEAL as subordinate extractors for 10 iterations of learning. We then compared the differences in performance between several pairs of methods to see the effects of coupling.

[2]CPL used the same corpus of sentences extracted from 200 million web pages as in Chapter 3.

4.4.1 Experimental Methodology

Input Ontology

We used the same ontology as the one used in Section 3.5.1. This allows us to directly compare CSEAL, SEAL, and MBL to the results from CPL and UPL.

Experimental Procedure

The experimental procedure for comparing two algorithms was the same as that used and described in Section 3.5.1. Each algorithm was run for 10 iterations of bootstrapping. Instances from each run and for each predicate were sampled and evaluated using Mechanical Turk.

CPL can reliably extract the proper case of an instance (i.e., capitalized vs. uncapitalized), but lists of items on the web often use arbitrary case conventions, so CSEAL cannot reliably extract the proper case of an instance. Because of this, our evaluation ignored case, and presented all instances to the evaluators in lower case.

Parameters for SEAL

In our experiments with CSEAL and SEAL, we used an implementation provided by the original authors of SEAL. SEAL was configured to subsample the provided examples 5 times for categories and 10 times for relations to mitigate the relatively higher sparsity of relations. SEAL downloaded up to 50 web pages for each search query using results from the Google search engine. Thus, the corpus for SEAL was the web as indexed by Google. The “minimum context length” for a wrapper was set to 2, which meant that each part of a wrapper needed to be at least 2 characters long.

4.4.2 Results for CSEAL and SEAL

We compared coupled and uncoupled methods for learning wrappers to extract lists of instances from semi-structured web pages:

	Estimated Precision (%)			Promoted Instances (#)		
Predicate	CSEAL	SEAL	MBL	CSEAL	SEAL	MBL
AcademicField	90	97	100	203	1000	181
Actor	100	97	100	1000	1000	380
Animal	90	70	97	144	974	307
Athlete	100	87	100	276	1000	555
AwardTrophyTournament	53	7	77	146	1000	79
BoardGame	70	77	90	126	1000	31
BodyPart	97	63	93	80	1000	61
Building	30	0	93	57	1000	14
Celebrity	100	100	97	72	747	514
CEO	100	77	100	322	1000	30
City	97	87	97	368	1000	603
Clothing	43	27	97	167	1000	102
Coach	100	83	100	619	1000	242
Company	100	100	97	245	1000	784
Conference	97	90	100	437	928	92
Country	97	37	93	130	1000	207
EconomicSector	100	10	77	34	1000	138
Emotion	87	60	83	183	1000	211
Food	97	80	100	89	1000	272
Furniture	57	57	90	215	1000	95
Hobby	77	50	90	77	1000	127
KitchenItem	88	13	100	8	960	2
Mammal	93	50	90	154	1000	169
Movie	97	100	100	566	1000	183
NewspaperCompany	60	97	100	1000	1000	241
Politician	97	37	100	30	1000	101
Product	-	77	70	0	999	127
ProductType	27	63	50	31	1000	159
Profession	-	57	93	0	1000	171
ProfessionalOrganization	100	77	87	58	1000	163
Reptile	90	27	100	149	1000	54
Room	33	7	100	12	643	3
Scientist	100	17	100	928	1000	130
Shape	7	7	85	28	733	26
Sport	63	83	73	225	1000	284
SportsEquipment	57	23	23	52	1000	174
SportsLeague	80	27	86	10	1000	14
SportsTeam	87	87	87	864	944	506
Stadium	53	63	90	944	1000	343
StateOrProvince	83	93	77	114	1000	161
Tool	93	90	97	713	1000	59
Trait	52	47	97	21	1000	44
University	100	90	93	961	1000	516
Vehicle	50	13	77	50	1000	98
Average	78	59	90	271	976	199
Weighted average	86	59	91			

Table 4.2: Estimated precision (%) and counts of promoted instances for each category for different algorithms. Precision for each predicate and method was estimated using 30 randomly sampled instances.

	Estimated Precision (%)			Promoted Instances (#)		
Predicate	CSEAL	SEAL	MBL	CSEAL	SEAL	MBL
CompanyAcquiredCompany	-	-	-	0	0	0
AthletePlaysForTeam	100	76	100	4	17	96
AthletePlaysInLeague	100	57	-	14	82	0
AthletePlaysSport	100	100	100	1	1	109
CEOOfCompany	-	100	100	0	1	1
CityLocatedInCountry	100	100	100	9	577	136
CityLocatedInState	100	93	100	34	537	54
CoachCoachesInLeague	0	-	-	1	0	0
CoachCoachesTeam	-	-	100	0	0	6
CompanyIsInEconomicSector	-	-	-	0	0	0
CompanyCompetesWithCompany	-	-	-	0	0	0
CompanyHasOfficeInCity	-	100	-	0	4	0
CompanyHasOfficeInCountry	-	-	-	0	0	0
CompanyHeadquarteredInCity	100	100	-	1	2	0
LeaguePlaysGamesInStadium	-	100	-	0	177	0
CompanyProducesProduct	-	-	100	0	0	8
ProductInstanceOfProductType	-	-	-	0	0	0
SportUsesSportsEquipment	100	87	33	5	15	6
StadiumLocatedInCity	77	70	90	200	554	56
StateHasCapitalCity	-	73	-	0	495	0
StateLocatedInCountry	100	97	100	46	653	61
TeamHasHomeStadium	100	100	100	179	106	92
TeamPlaysAgainstTeam	-	-	-	0	0	0
TeamHasHomeCity	-	93	100	0	29	11
TeamPlaysInLeague	100	100	100	104	749	23
TeamPlaysSport	100	100	100	30	30	37
TeamWonAwardTrophyTournament	-	-	-	0	0	0
Average	91	91	95	23	149	26
Weighted Average	92	90	99			

Table 4.3: Estimated precision (%) and counts of promoted instances for each relation for different algorithms. Precision for each predicate and method was estimated using 30 randomly sampled instances.

	CSEAL vs. SEAL		MBL vs. CSEAL		MBL vs. CPL	
Predicate	SEAL	CSEAL	CSEAL	MBL	CPL	MBL
AcademicField	97	93	93	100	90	100
Actor	97	97	100	100	100	100
Animal	100	83	93	100	97	93
Athlete	77	97	100	100	97	100
AwardTrophyTournament	33	57	77	67	73	67
BoardGame	87	87	97	90	80	100
BodyPart	80	90	93	87	63	90
Building	63	37	64	93	93	93
Celebrity	100	100	100	100	100	100
CEO	70	100	93	100	33	100
City	97	100	100	100	100	97
Clothing	57	57	47	90	100	100
Coach	60	83	93	100	97	100
Company	100	97	100	100	100	100
Conference	97	97	97	100	90	97
Country	100	97	97	97	97	90
EconomicSector	97	100	100	97	90	77
Emotion	73	70	90	83	70	87
Food	100	100	97	100	97	100
Furniture	63	70	57	93	93	97
Hobby	63	73	77	100	77	100
KitchenItem	88	88	100	100	50	100
Mammal	57	87	90	83	90	90
Movie	93	100	90	100	93	100
NewspaperCompany	97	63	67	93	97	90
Politician	80	97	97	100	90	100
Product	-	-	-	-	77	67
ProductType	50	30	30	43	83	37
Profession	-	-	-	-	73	100
ProfessionalOrganization	97	97	97	100	83	93
Reptile	57	77	90	97	95	95
Room	42	33	33	100	100	100
Scientist	10	97	97	100	100	100
Shape	14	7	8	85	73	85
Sport	97	60	83	63	87	87
SportsEquipment	53	60	57	57	20	43
SportsLeague	70	80	80	80	100	82
SportsTeam	87	100	97	83	97	100
Stadium	70	63	77	90	97	97
StateOrProvince	80	63	73	87	90	77
Tool	87	97	100	97	43	93
Trait	95	52	52	100	73	97
University	90	87	97	100	97	97
Vehicle	67	53	53	90	83	83
Average	76	78	82	92	85	91

Table 4.4: Comparing different pairs of algorithms at equivalent levels of recall, this table gives the estimated precision (%) of promoted instances for each category when only considering instances using the "Minimum Recall" threshold. Each precision was estimated using 30 randomly sampled instances.

	CSEAL vs. SEAL		MBL vs. CSEAL		MBL vs. CPL	
Predicate	SEAL	CSEAL	CSEAL	MBL	CPL	MBL
AthletePlaysForTeam	50	100	100	100	100	100
AthletePlaysInLeague	93	100	-	-	-	-
AthletePlaysSport	100	100	100	100	100	100
CEOOfCompany	-	-	-	-	100	100
CityLocatedInCountry	100	100	100	100	87	100
CityLocatedInState	100	100	100	100	100	100
CoachCoachesTeam	-	-	-	-	100	100
CompanyHeadquarteredInCity	100	100	-	-	-	-
CompanyProducesProduct	-	-	-	-	100	100
SportUsesSportsEquipment	60	100	100	20	17	33
StadiumLocatedInCity	90	67	77	90	100	100
StateLocatedInCountry	93	100	100	100	97	100
TeamHasHomeStadium	100	97	97	100	97	100
TeamPlaysInLeague	100	100	100	100	100	100
TeamPlaysSport	100	100	100	100	-	-
Average	91	97	97	91	92	94

Table 4.5: Comparing different pairs of algorithms at equivalent levels of recall, this table gives the estimated precision (%) of promoted instances for each relation when only considering instances using the "Minimum Recall" threshold. Precision for each predicate and method was estimated using 30 randomly sampled instances.

	All Promotions		Minimum Recall	
Comparison	Wins	p-value	Wins	p-value
CSEAL vs. SEAL	36 vs. 15	0.00460	22 vs. 18	0.636
MBL vs. CPL	34 vs. 18	0.0365	25 vs. 10	0.0167
MBL vs. CSEAL	31 vs. 14	0.0161	26 vs. 10	0.0113

Table 4.6: Various pairs of methods compared based on the estimated precision of all promotions for each predicate (All Promotions) and the estimated precision of the instances promoted cut off at the minimum recall out of the pair for each predicate (Minimum Recall). Wins record how many predicates had superior estimated precision for each method, and the p-value according to a sign test is given. All results are statistically significant at the 5% level except for CSEAL vs. SEAL at minimum recall.

- CSEAL: The algorithm as described in Section 4.2.3.
- SEAL: This method uses the implementation of SEAL provided by the authors of SEAL and described in Section 4.2.2. As in the UPL method, it does not couple the learning of predicates using mutual-exclusion constraints or type checking.

Table 4.2 gives estimates of the precision of promoted instances for each category for CSEAL and SEAL, as well as the number of promoted instances for each category after 10 iterations. The "Average" row averages across all predicates for which instances were promoted. The "Weighted Average" is an estimate of the instance-level precision across all predicates obtained by weighting the estimated precision for each predicate by the number of instances promoted for that predicate. Table 4.3 gives this information for each relation, as well. Across all categories, CSEAL has higher average estimated precision than SEAL. For relations, there is no difference in average estimated precision. These results suggest that coupling using type checking and mutual exclusion reduces the error rates of the learned extractors for categories, but not relations.

Tables 4.4 and 4.5 compare CSEAL and SEAL at equivalent levels of recall for categories and relations, respectively (see Section 3.5.1 for a description of the "Minimum Recall" method of comparing predicates at equivalent recall). CSEAL improves over SEAL when looking at the macro-average precision for both categories and relations.

Table 4.6 uses Sign Tests to compare CSEAL vs. SEAL for the estimated precision of all promoted instances for each predicate, as well as the “minimum recall” sample discussed in Section 3.5.1. CSEAL performs significantly better than SEAL with respect to the estimated precision of all promotions, but is not significantly better when thresholding recall to the minimum recall for each predicate.

4.4.3 Results for Meta-Bootstrap Learner

Tables 4.2 and 4.3 give estimates of the precision of promoted instances for each predicate for MBL after 10 iterations. Across both relations and categories, MBL has the highest estimated precision of promoted instances out of all of the algorithms considered, which indicates that adding the multi-view-agreement constraint results in further avoidance of semantic drift.

Tables 4.4 and 4.5 compare MBL to CSEAL and CPL equivalent levels of recall for categories and relations. For categories, MBL clearly improves over both CSEAL and CPL when looking at macro-average precision. For relations, MBL is worse than CSEAL, because “SportUsesSportsEquipment” fares so poorly (due to bad type-checking of sports equipment).

Table 4.6 gives sign test results for comparing MBL vs. CPL and MBL vs. CSEAL, which allows us to judge whether or not MBL improves over its subordinate algorithms. All sign tests show statistically significant differences: MBL is superior to both CPL and CSEAL when comparing both the estimated precision of all promoted instances as well as the estimated precision of promoted instances at the minimum recall of either method. This suggests that coupling CPL and CSEAL with a multi-view coupling constraint yields more accurate learning than either method used alone.

4.4.4 Discussion

We saw that CSEAL has higher estimated precision than SEAL for categories, but not relations. We believe that this is because when a set of relation instances occur on a web page in a consistent context, it is generally safe to assume that instances extracted from that web page are correct instances of the target relation. There are two reasons to think that this is the case. First, the queries are more restrictive; documents are found using pairs of arguments rather than single arguments as in the case of categories. Second, wrappers for relations are more restrictive than those for categories, since they have an infix string in addition to a prefix and suffix string. Categories are more likely to have a set of instances occur on a page without it being a valid list or table, and category templates are less restrictive. This appears to be why filtering using mutual exclusion improves precision for categories significantly.

One of the biggest challenges in applying bootstrap learning algorithms is determining when to stop the bootstrapping process. Ideally, an algorithm would be able to respect the boundaries of a closed set. In this respect, the results for the Country category for MBL are particularly compelling. MBL promoted 207 instances of countries with an estimated precision of 93%. CSEAL promoted 130 instances with an estimated precision of 97%. Without coupling (UPL, SEAL), Country performs poorly, drifting into a more general Location category.

While MBL has the highest estimated precision overall, its recall is relatively low when compared to the other methods, particularly for relations. This is because there are many relations for which only one of CSEAL and CPL has significant recall, and for these relations MBL will not promote any instances. MBL demonstrates that when multiple extraction methods that make independent errors extract an instance, we can be confident. In the next chapter, we consider a more sophisticated strategy that promotes instances when a single method is very confident in that extraction, as well as when multiple methods extract that instance. This leads to higher recall while maintaining high precision.

4.4.5 Supplementary Online Materials

In addition to the supplementary online materials from the evaluation in the previous chapter, several different types of materials from the current evaluation are posted online at `http://rtw.ml.cmu.edu/acarlson_thesis`:

- All instances promoted by MBL, CSEAL, and SEAL.
- All textual patterns promoted by pattern learning in the MBL experiments.
- Browseable knowledge bases in XML format of all promoted instances and candidate instances from the runs of MBL and CSEAL, with patterns and URLs that extracted each instance.
- All judgments obtained from Mechanical Turk used in estimates of precision.

4.5 Conclusion

In Chapter 3, we saw that coupled semi-supervised learning can improve learning of textual extraction patterns. In this chapter, we showed that the same ideas can improve an off-the-shelf, state-of-the-art system that uses a different extraction method, wrapper induction from semi-structured documents. This demonstrates that the ideas of coupled semi-supervised learning may have broad applicability to a variety of learning problems. We also showed that a different type of coupling, multi-view coupling, can improve the precision of semi-supervised learning by combining the predictions from two different extraction methods.

Chapter 5

Scaling Up: More Predicates, More Extraction Methods

Abstract

The results in previous chapters lead to two questions: (1) Can we scale up the number and variety of predicates in our ontology and still maintain high precision with coupled semi-supervised learning methods? and (2) Should we consider adding additional extraction methods to CPL and CSEAL, coupling more than two techniques together? In this chapter, we explore these questions by learning to extract over 150 predicates (compared to 71 before), and by using four different extraction methods. We first describe a general architecture that can exploit many different extraction methods. The architecture uses coupled semi-supervised learning methods, an ensemble of varied knowledge extraction methods, and a flexible knowledge base that allows the integration of the outputs of those methods. We also discuss design principles for implementing this architecture. We then describe a prototype implementation of our architecture, called *Multi-Extractor Coupler* (MEC). With an extended ontology of 123 categories and 55 relations, MEC has learned to extract a knowledge base containing over 242,000 beliefs with an estimated precision of 74%. Analysis of the results shows that each of the four extraction methods contributes positively to these results.

5.1 Introduction

In Chapter 3 and Chapter 4, we saw how coupled semi-supervised learning forestalls semantic drift and improves the precision of pattern-based information extractors and wrapper inducers for semi-structured web pages. We also saw that coupling the training of these two methods by only promoting instances extracted by both methods can lead to very high precision. The results from these chapters lead to the questions: (1) *Can we scale up the number and variety of predicates in our ontology and still maintain high precision with coupled semi-supervised learning methods?* and (2) *Should we consider adding additional extraction methods to CPL and CSEAL, coupling more than two techniques together?*

To answer the first question, we scale up our ontology to 123 categories and 55 relations (compared to 44 categories and 27 relations in previous chapters). This provides an opportunity to see how well coupled semi-supervised learning methods can work on a larger number and variety of predicates.

To answer the second question, we first describe a general architecture that can exploit many different extraction methods. The architecture uses coupled semi-supervised learning methods, an ensemble of varied knowledge extraction methods, and a flexible knowledge base that allows the integration of the outputs of those methods. We also discuss design principles for implementing this architecture.

We then describe a prototype implementation of our architecture, called *Multi-Extractor Coupler* (MEC). At present, MEC acquires the two types of knowledge discussed earlier in this thesis: (1) knowledge about what noun phrases refer to some specified semantic *categories*, such as cities, companies, and universities, and (2) knowledge about what pairs of noun phrases satisfy some specified semantic *relations*, such as hasOfficesIn(organization, location). MEC learns to acquire these two types of knowledge in four ways: it learns free-form text patterns for extracting this knowledge from sentences on the web, it learns to extract this knowledge from semi-structured web data such as tables and lists, it learns morphological regularities of instances of categories, and it learns probabilistic horn clause rules that enable it to infer new instances of relations from other relation instances that it has already learned. Compared to MBL from Chapter 4, MEC uses a different strategy for combining evidence from multiple extractors: candidate facts that have high-confidence from a single source are promoted, and lower-confidence candidates are also promoted if they have been proposed by multiple sources independently. MBL only promoted instances that satisfied the latter condition.

Finally, we present experiments showing that our implementation of MEC, given an initial seed ontology defining 123 categories and 55 relations and left to run for six days, populates this ontology over 242,000 new facts with estimated precision of 74%. This is well over an order of magnitude more facts than those learned by MBL in the previous chapter, reflecting MEC's ability to leverage several different learning methods and a larger ontology to achieve higher recall while still maintaining high precision. Analysis of the results shows that each of the four extraction methods contributes positively to these results.

The material presented in this chapter was originally presented by Carlson *et al.* [2010a] and is based on research done in collaboration with co-authors.

5.2 Approach

Our approach to coupling the learning of multiple extractors is organized around a shared knowledge base (KB) that is incrementally and continuously grown and used by a collection of learning subsystem components that implement complementary knowledge extraction methods. The starting KB defines an ontology (a collection of predicates defining categories and relations), and a handful of seed examples for each predicate in this ontology (e.g., a dozen example cities). The goal of our approach is to continuously grow this KB.

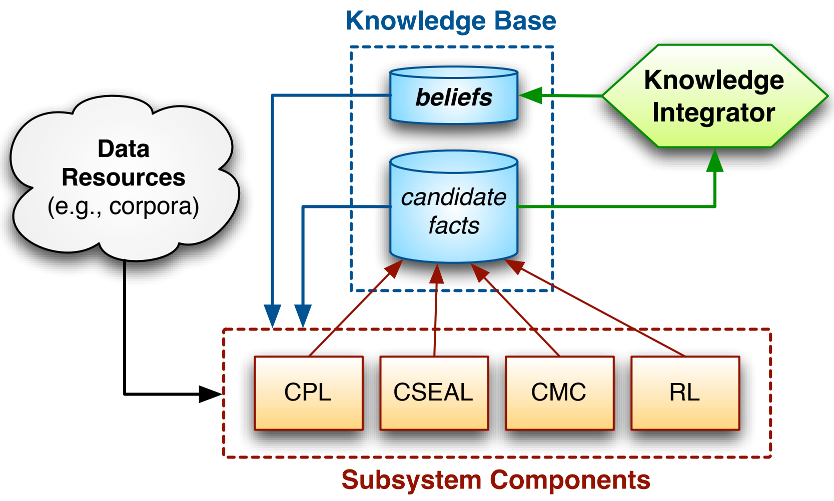


Figure 5.1: Our Multi-Extractor Coupler (MEC) architecture. See "Approach" for an overview of the approach implemented in MEC, and "Implementation" for subsystem details.

Category and relation instances added to the KB are partitioned into *candidate facts* and *beliefs*. The subsystem components can read from the KB and consult other external resources (e.g., corpora or the Internet), and then propose new candidate facts. Components supply a probability for each candidate and a summary of the source evidence supporting it. The Knowledge Integrator (KI) examines these candidate facts and promotes the most strongly supported of these to belief status. This flow of processing is depicted in Figure 5.1.

In our initial implementation, as detailed in the Implementation section, our approach operates iteratively. On each iteration, subsystem components are run to completion given the current KB and each outputs its proposed candidate facts. The KI then makes its decisions on which candidate facts to promote. The KB grows, and this provides stronger training information to each component, and this in turn allows each component to learn to read better. In this way, our approach can be seen as implementing a coupled, semi-supervised learning method in which multiple components learn and share complementary types of knowledge, overseen by the KI.

This kind of iterative learning approach can suffer if labeling errors accumulate. To help mitigate this issue, we envision that the system should be able to interact with a human for 10–15 minutes each day, to help the learner stay "on track," though the experiments reported here make limited use of such human input.[1]

This approach was designed as a step toward *never-ending learning*, a paradigm where a system can continually learn to improve its abilities in a performance task. In the future, we intend to involve greater human interaction, and intelligent planning that could, for example, decide to invoke specific extractors targetted toward specific predicates, or decide to extend its ontology with new predicates.

The following design principles are important in implementing our approach to coupling the learning of many extractors:

- Use subsystem components that make uncorrelated errors. When multiple components that make uncorrelated errors propose the same candidate fact, we can typically be quite confident in that belief.

- Learn multiple types of inter-related knowledge. For example, we use one component that learns to extract predicate instances from text resources, and another which learns to infer relation instances from other beliefs in the KB. This provides multiple, independent sources of the same types of beliefs.

- Use coupled semi-supervised learning methods to leverage constraints between predicates being learned. To provide opportunities for coupling, arrange categories and relations into a taxonomy, and declare most categories and relations to be mutually exclusive. Additionally, specify the expected category of each relation argument to enable type-checking. Subsystem components and the KI can benefit from methods that leverage coupling.

- Distinguish high-confidence beliefs in the KB from lower-confidence candidates, and retain

[1]A few minutes were spent by a human judging proposed rules 6 times during our experiments, as described in Section 5.5.1.

source justifications for each belief.

- Use a uniform KB representation to capture candidate facts and promoted beliefs of all types, and use associated inference and learning mechanisms that can operate on this shared representation.

5.3 Related Work

AI has a long history of research on autonomous agents, problem solving, and learning, e.g., SOAR [Laird *et al.*, 1987], PRODIGY [Carbonell *et al.*, 1991], EURISKO [Lenat, 1983], ACT-R [Anderson *et al.*, 2004], and ICARUS [Langley *et al.*, 1991]. In comparison, our focus in this chapter is on semi-supervised learning for information extraction, with less focus on problem-solving search. Nevertheless, earlier work provides a variety of design principles upon which we have drawn. For example, the role of the KB in our approach is similar to the role of the "blackboard" in early systems for speech recognition [Erman *et al.*, 1980], and the frame-based representation of our KB is a reimplementation of the THEO system [Mitchell *et al.*, 1991] which was originally designed to support integrated representation, inference and learning. There is also previous research on life-long learning, such as Thrun and Mitchell [1995], which focuses on using previous learning tasks to bias new learning tasks. Banko and Etzioni [2007] consider a lifelong learning setting where an agent is building a *theory* of a domain. Their work explores different strategies for deciding which of many possible learning tasks to tackle next. Our implementation operates on all predicates in each iteration of learning; as we consider methods of focusing attention on specific predicates in the future, Banko and Etzioni's work could be instructive.

Pennacchiotti and Pantel [2009] present a framework for combining the outputs of an ensemble of extraction methods, which they call "Ensemble Semantics." Multiple extraction systems provide candidate category instances, which are then ranked using a learning function that uses features from many different sources (e.g., query logs, Wikipedia). Their approach uses a more sophisticated ranking method than ours, but is not iterative. Thus, their ideas are complementary to our work, as we could use their ranking method as part of our general approach.

5.4 Implementation

We have implemented a preliminary version of our approach. We call this implementation *Multi-Extractor Coupler* (MEC). MEC uses four subsystem components (Figure 5.1):

- *Coupled Pattern Learner* (CPL): A free-text extractor which learns and uses contextual patterns like "mayor of X" and "X plays for Y" to extract instances of categories and relations. CPL uses co-occurrence statistics between noun phrases and contextual patterns (both defined using part-of-speech tag sequences) to learn extraction patterns for each predicate of interest and then uses those patterns to find additional instances of each predicate. Relationships between predicates are used to filter out patterns that are too general. CPL is described in detail in Chapter 3. Probabilities of candidate instances extracted by CPL are heuristically assigned using the formula $1-0.5^c$, where c is the number of promoted patterns that extract a candidate. In our experiments, CPL was given as input a corpus of 2 billion sentences, which was generated by using the OpenNLP package[2] to extract, tokenize, and part-of-speech tag sentences from the 500 million web page ClueWeb09 data set [Callan and Hoy, 2009]. CPL was written by Andrew Carlson and Justin Betteridge. For more details, see Chapter 3.

- *Coupled SEAL* (CSEAL): A semi-structured extractor which queries the Internet with sets of beliefs from each category or relation, and then mines lists and tables to extract novel instances of the corresponding predicate. CSEAL uses mutual exclusion relationships to provide negative examples, which are used to filter out overly general lists and tables. CSEAL

[2] http://opennlp.sourceforge.net

is described in Chapter 4. Given a set of seed instances, CSEAL performs queries by sub-sampling beliefs from the KB and using these sampled seeds in a query. CSEAL was configured to issue 5 queries for each category of interest and 10 queries for each relation of interest, and to fetch 50 web pages per query. Candidate facts extracted by CSEAL are assigned probabilities using the same method as for CPL, except that c is the number of unfiltered wrappers that extract an instance. SEAL was written by Richard Wang and William Cohen. CSEAL was written by Andrew Carlson and Estevam Hruschka. For more details, see Chapter 4.

- *Coupled Morphological Classifier* (CMC): A set of binary L_2-regularized logistic regression models—one per category—which classify noun phrases based on various morphological features (words, capitalization, affixes, parts-of-speech, etc.). Beliefs from the KB are used as training instances, but at each iteration CMC is restricted to predicates which have at least 100 positives. As with CSEAL, mutual exclusion relationships are used to identify negative instances. CMC examines candidate facts proposed by other components, and classifies up to 30 new beliefs per predicate per iteration, with a minimum posterior probability of 0.75. These heuristic measures help to ensure high precision, generating increased support for existing candidates and enforcing morphological constraints on other subsystems. CMC was implemented in Java by Burr Settles.

- *Rule Learner* (RL): A first-order relational learning algorithm similar to FOIL [Quinlan and Cameron-Jones, 1993], which learns probabilistic Horn clauses. These learned rules are used to infer new relation instances from other relation instances that are already in the KB. RL was implemented in Matlab by Tom Mitchell.

Our implementation of the Knowledge Integrator (KI) promotes candidate facts suggested by the other components to the status of beliefs, using a hard-coded, intuitive strategy. Candidate facts that have high-confidence from a single source (those with posterior > 0.9) are promoted, and lower-confidence candidates are promoted if they have been proposed by multiple sources independently. KI exploits relationships between predicates by respecting mutual exclusion and type checking information. In particular, candidate category instances are not promoted if they already belong to a mutually exclusive category, and relation instances are not promoted unless their arguments are at least candidates for the appropriate category types (and are not already believed to be instances of a mutually exclusive category). In our current implementation, once a candidate fact is promoted as a belief, it is never demoted. The KI is configured to promote up to 250 instances per predicate per iteration, but this threshold was rarely hit in our experiments.

The KB in MEC is a reimplementation of the THEO frame-based representation [Mitchell *et al.*, 1991] based on Tokyo Cabinet[3], a fast, lightweight key/value store. The KB can handle many millions of values on a single machine.

5.5 Experimental Evaluation

We conducted an experimental evaluation to explore the following questions:

1. Can MEC learn to populate many different categories (100+) and relations (50+) for 20+ iterations of learning and maintain high precision?

2. How much do the different components contribute to the promoted beliefs held by MEC?

5.5.1 Methodology

The input ontology used in our experiments included 123 categories each with 10–15 seed instances and 5 seed patterns for CPL (derived from Hearst patterns [Hearst, 1992]). Categories

[3] `http://1978th.net/tokyocabinet`

included locations (e.g., mountains, lakes, cities, museums), people (e.g., scientists, writers, politicians, musicians), animals (e.g., reptiles, birds, mammals), organizations (e.g., companies, universities, web sites, sports teams), and others. 55 relations were included, also with 10–15 seed instances and 5 negative instances each (typically generated by permuting the arguments of seed instances). Relations captured relationships between the different categories (e.g., teamPlaysSport, bookWriter, companyProducesProduct).

In our experiments, CPL, CSEAL, and CMC ran once per iteration. RL was run after each batch of 10 iterations, and the proposed output rules were filtered by a human. Manual approval of these rules took only a few minutes.

To estimate the precision of the beliefs in the KB produced by MEC, beliefs from the final KB were randomly sampled and evaluated by several human judges. Cases of disagreement were discussed in detail, with final decisions made by another judge. Facts which were once true but are not currently (e.g., a former coach of a sports team) were considered to be correct for this evaluation, as MEC does not currently deal with temporal scope in its beliefs. Spurious adjectives (e.g., "today's Chicago Tribune") were allowed, but rare.

5.5.2 Results

After running for 67 days, MEC completed 66 iterations of execution. 242,453 beliefs were promoted across all predicates, 95% of which were instances of categories and 5% of relations. Example beliefs from a variety of predicates, along with the source components that extracted them, are shown in Table 5.1.

Promotion Rates

Following an initial burst of almost 10,000 beliefs promoted during the first iteration, MEC continued to promote a few thousand more on every successive iteration, indicating strong potential

Predicate	Instance	Source(s)
ethnicGroup	Cubans	CSEAL
arthropod	spruce beetles	CPL, CSEAL
female	Kate Mara	CPL, CMC
sport	BMX bicycling	CSEAL, CMC
profession	legal assistants	CPL
magazine	Thrasher	CPL
bird	Buff-throated Warbler	CSEAL
river	Fording River	CPL, CMC
mediaType	chemistry books	CPL, CMC
cityInState	(troy, Michigan)	CSEAL
musicArtistGenre	(Nirvana, Grunge)	CPL
tvStationInCity	(WLS-TV, Chicago)	CPL, CSEAL
sportUsesEquip	(soccer, balls)	CPL
athleteInLeague	(Dan Fouts, NFL)	RL
starredIn	(Will Smith, Seven Pounds)	CPL
productType	(Acrobat Reader, FILE)	CPL
athletePlaysSport	(scott shields, baseball)	RL
cityInCountry	(Dublin Airport, Ireland)	CPL

Table 5.1: Example beliefs promoted by MEC.

to learn more if it were left to run for a longer time. Figure 5.2 shows different views of the promotion activity of MEC over time. The left-hand figure shows overall numbers of promotions for categories and relations in each iteration. Category instances are promoted fairly steadily, while relation instance promotions are spiky. This is mainly because the RL component only runs every 10 iterations, and is responsible for many of the relation promotions. The right-hand figures are stacked bar plots showing the proportion of predicates with various levels of promotion activity during different spans of iterations. These plots show that instances are promoted for many different categories and relations during the whole run of MEC.

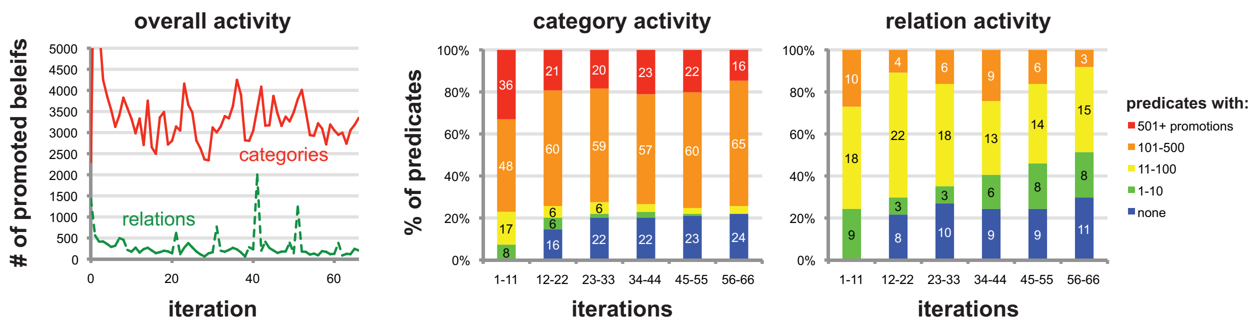


Figure 5.2: Promotion activity for beliefs over time. *Left:* The number of beliefs promoted for all category and relation predicates in each iteration. Periodic spikes among relation predicates occur every 10 iterations after the RL component runs. *Center and Right:* Stacked bar plots detailing the proportion of predicates (and counts of predicates, shown inside the bars) at various levels of promotion activity over time for categories and relations. Note that, while some predicates become "dormant" early on, the majority continue to show healthy levels of promotion activity even in later iterations of learning.

Estimates of Precision

To estimate the precision of beliefs promoted during various stages of execution, we considered three time periods: iterations 1–22, iterations 23–44, and iterations 45–66. For each of these time periods, we uniformly sampled 100 beliefs promoted during those periods and judged their correctness. The results are shown in Table 5.2. During the three periods, the promotion rates are very similar, with between 76,000 and 89,000 instances promoted. There is a downward trend in estimated precision, going from 90% to 71% to 57%. Taking a weighted average of these three estimates of precision based on numbers of promotions, the overall estimated precision across all 66 iterations is 74%.

Only a few items were debated by the judges: examples are "right posterior," which was judged to not refer to a body part, and "green leafy salad," which was judged acceptable as a type of vegetable. "Proceedings" was promoted as a publication, which we considered incorrect (it was most likely due to noun-phrase segmentation errors within CPL). Two errors were due

Iterations	Estimated Precision (%)	# Promotions
1–22	90	88,502
23–44	71	77,835
45–66	57	76,116

Table 5.2: Estimates of precision (from 100 sampled beliefs) and numbers of promoted beliefs across all predicates during iterations 1–22, 23–44, and 45–66. Note that the estimates of precision only consider beliefs promoted during a time period and ignore beliefs promoted earlier.

to languages ("Klingon Language" and "Mandarin Chinese language") being promoted as ethnic groups. ("Southwest", "San Diego") was labeled as an incorrect instance of the hasOfficesIn relation, since Southwest Airlines does not have an official corporate office there. Many system errors were subtle; one might expect a non-native reader of English to make similar mistakes.

To estimate precision at the predicate level, we randomly chose 7 categories and 7 relations which had at least 10 promoted instances. For each chosen predicate, we sampled 25 beliefs from iterations 1–22, 23–44, and 45–66, and judged their correctness. Table 5.3 shows these predicates and, for each time period, the estimates of precision and the number of beliefs promoted. Most predicates are very accurate, with precision exceeding 90%. Two predicates in particular, cardGame and productType, fare much worse. The cardGame category seems to suffer from the abundance of web spam related to casino and card games, which results in parsing errors and other problems. As a result of this noise, MEC ends up extracting strings of adjectives and nouns like "deposit casino bonuses free online list" as incorrect instances of cardGame. Most errors for the productType relation came from associating product names with more general nouns that are somehow related to the product but do not correctly indicate what kind of thing the product is, e.g., ("Microsoft Office", "PC"). Some of these productType beliefs were debated by the judges, but were ultimately labeled incorrect, e.g., ("Photoshop", "graphics"). In our ontology, the category for the second argument of productType is a general "item" super-category in the hierarchy; we posit that a more specific "product type" category might lead to more restrictive type checking.

Analysis of Errors

To gain a deeper understanding of the errors made by MEC during a long run, a collaborator looked at the instances promoted by MEC after 100 iterations of execution had been finished. The annotator had two ways of making judgments: (1) Promoted instances could be explicitly marked as incorrect, and (2) Predicates could be marked as untrustworthy after a specific point in time, in which case all promotions after that were discarded.

During the run, 320,892 category instances were promoted. 5,762 were explicitly marked as errors, while another 76,146 were marked to be discarded because the category had diverged. Out of 24,132 relation instances, 247 were explicitly marked wrong, and 456 were marked to be discarded due to divergence.

Appendix B lists many selected predicates with an informal description of major trends in errors for each predicate. From this analysis, we see several trends:

- **Near-Miss Negative Examples are Vital**: Predicates need good sources of negative examples through mutual-exclusion coupling constraints. When they are not available, the boundaries of a predicate are poorly defined, and such predicates tend to diverge. For example, the relations “teamPlaysInLeague” and “cityCapitalOfState” lack good negative examples of teams that play football that are not in the NFL, and of states that are in cities but are not capital cities. The category “ethnicGroup” diverged to include natural languages because there was no “naturalLanguage” category.
- **Every Component Fails Sometime**: In the appendix, each of the four components used by MEC causes divergence for at least one predicate. CPL learns patterns that lead to lots of newspapers being incorrectly based in New York. CSEAL learns noisy segments of text like “bivalves are filter feeders” for the “invertebrate” category. CMC learns a high weight for the word “coffee” for the “beverage” category which leads to “metal coffee” being promoted. Finally, RL incorrectly concludes that every sports team that plays “football” plays in the league

"NFL". In many of these cases, the components were confident enough to cause MEC to promote these instances without corraboration from another source. Since MEC will promote high-confidence candidates from a single source, these errors are promoted. A possible way to reduce these errors would be to use other components to assess these instances to get a "second opinion."

- **Noisy Noun Phrases are a Problem**: The segmentation of noun phrases in CPL fails often, leading to "Life Catering" being chopped off from the end of the full phrases "Spice of Life Catering." CSEAL learns templates that extract instances like "what plants grow in the mountains". It is worth investigating adding a filter to try to remove such noisy instances.
- **Correlated Errors Happen**: Our design principle of choosing components that tend to make uncorrelated errors is sometimes violated in the results. For example, while CMC learns that the word "sauce" is a good feature for the "condiment" category, if CPL and CSEAL were perfect, this would not be a problem. However, CPL extracts "sauce teaspoon" as a candidate instance, and these correlated errors lead to incorrect promotions.

These points lead us to conclude that we should try to filter out noisy NPs, design ontologies so that every predicate has a good source of negative evidence, and refine our Knowledge Integrator to deal with the fact that high-confidence single-source promotions can be incorrect. Also, we should get a human involved in the loop in MEC; the regularity of the errors described in Appendix B suggest that a few negative examples for a predicate at the right time could keep learning on track.

Contributions of Each Learning Component

As described in the Implementation section, MEC uses a Knowledge Integrator which promotes high-confidence single-source candidate facts, as well as candidate facts with multiple lower-confidence sources. Figure 5.3 illustrates the impact of each component within this integration strategy. Each component is shown containing a count which is the number of beliefs that were

	Estimated Precision			# Promotions		
Predicate	1–22	23–44	45–66	1–22	23–44	45–66
cardGame	40	20	0	584	552	2,472
city	92	80	96	4,311	3,362	1,002
magazine	96	68	80	1,235	788	664
recordLabel	100	100	100	1,384	890	748
restaurant	96	88	92	242	568	523
scientist	96	100	100	768	1	404
vertebrate	100	100	96	1,196	1,362	714
athletePlaysForTeam	100	100	100	113	304	39
ceoOfCompany	100	100	100	82	8	9
coachesTeam	100	100	100	196	121	12
productType	28	44	20	35	156	195
teamPlaysAgainstTeam	96	100	100	283	553	232
teamPlaysSport	100	100	86	79	158	14
teamWonTrophy	88	72	44	119	104	174

Table 5.3: For selected categories (top) and relations (bottom), estimates of precision (from 25 sampled beliefs) and counts for beliefs promoted during iterations 1–22, 23–44, and 45–66.

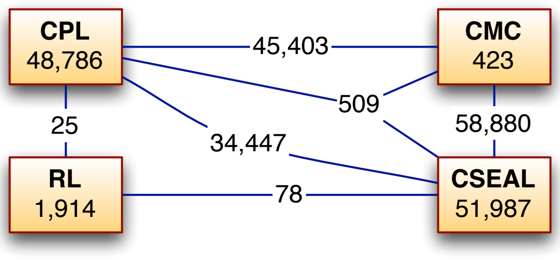


Figure 5.3: Source counts for beliefs promoted by MEC after 66 iterations. Numbers inside nodes indicate the number of beliefs promoted based solely on that component. Numbers on edges indicate beliefs promoted based on evidence from multiple components.

promoted based on that source alone having high confidence in that belief. Lines connecting components are labeled with counts that are the number of beliefs promoted based on those components each having some degree of confidence in that candidate. CPL and CSEAL each were responsible for many promoted beliefs on their own. However, more than half of the beliefs promoted by KI were based on multiple sources of evidence. While RL was not responsible for many promoted beliefs, those that it did propose with high confidence appear to be largely independent from those of the other components.

RL learned an average of 66.5 novel rules per iteration, of which 92% were approved. 12% of the approved rules implied at least one candidate instance that had not yet been implied by another rule, and those rules implied an average of 69.5 such instances.

To give a sense of what is being learned by the different components used in MEC, we provide examples for each component. Table 5.4 shows contextual patterns learned by CPL. Table 5.5 shows web page wrappers learned by CSEAL. Example weights from the logistic regression classifiers learned by CMC are shown in Table 5.6. Finally, example rules induced by RL are shown in Table 5.7.

Predicate	Pattern
emotion	hearts full of X
beverage	cup of aromatic X
newspaper	op-ed page of X
teamPlaysInLeague	X ranks second in Y
bookAuthor	Y classic X

Table 5.4: Example free-text patterns learned by CPL. X and Y represent placeholders for noun phrases to be extracted.

Predicate	Web URL	Extraction Template
academicField	http://scholendow.ais.msu.edu/student/ScholSearch.Asp	`&nbsp;`[X] `-`
bird	http://www.michaelforsberg.com/stock.html	`<option>`[X]`</option>`
bookAuthor	http://lifebehindthecurve.com/	`</li> <li>`[X] `by` [Y] `&#8211;`

Table 5.5: Examples of web page extraction templates learned by the CSEAL subsystem. [X] and [Y] represent placeholders for instances to be extracted (categories have only one placeholder; relations have two).

Predicate	Feature	Weight
mountain	LAST=peak	1.791
mountain	LAST=mountain	1.093
mountain	FIRST=mountain	-0.875
musicArtist	LAST=band	1.853
musicArtist	POS=DT_NNS	1.412
musicArtist	POS=DT_JJ_NN	-0.807
newspaper	LAST=sun	1.330
newspaper	LAST=press	1.276
newspaper	LAST=university	-0.318
university	LAST=college	2.076
university	PREFIX=uc	1.999
university	LAST=university	1.745
university	FIRST=college	-1.381
visualArtMovement	SUFFIX=ism	1.282
visualArtMovement	PREFIX=journ	-0.234
visualArtMovement	PREFIX=budd	-0.253

Table 5.6: Example feature weights induced by the morphology classifier. Positive and negative weights indicate positive and negative impacts on predicted probabilities, respectively. Note that "mountain" and "college" have different weights when they begin or end an instance. The learned model uses part-of-speech features to identify typical music group names (e.g., The Beatles, The Ramones), as well as prefixes to disambiguate art movements from, say, academic fields and religions.

Probability	Consequent	Antecedents
0.95	athletePlaysSport(X, basketball)	$\Leftarrow$ athleteInLeague(X, NBA)
0.91	teamPlaysInLeague(X, NHL)	$\Leftarrow$ teamWonTrophy(X, Stanley Cup)
0.90	athleteInLeague(X, Y)	$\Leftarrow$ athletePlaysForTeam(X, Z), teamPlaysInLeague(Z, Y)
0.88	cityInState(X, Y)	$\Leftarrow$ cityCapitalOfState(X, Y), cityInCountry(X, USA)
† 0.62	newspaperInCity(X, New York)	$\Leftarrow$ companyEconSect(X, media), generalizations(X, blog)

Table 5.7: Example horn clauses induced by the rule learner. Probabilities indicate the conditional probability that the literal to the left of $\Leftarrow$ is true given that the literals to the right are satisfied. Each rule captures an empirical regularity among the relations mentioned by the rule. The rule marked with † was rejected during human inspection.

Supplementary Online Materials

Several types of supplementary materials from our evaluation are posted online[4], including: (1) all promoted instances, (2) all categories, relations, and seed instances, (see Section 1.4.1 for a description of all of the information specified in the ontology) (3) all labeled instances sampled for estimating precision, (4) all patterns promoted by CPL, and (5) all rules learned by RL.

5.5.3 Discussion

These results demonstrate that coupled semi-supervised learning can scale to learn hundreds of thousands of facts for hundreds of predicates while maintaining 74% precision. They also demonstrate that many different extraction techniques can be coupled together in a positive way; we saw unique instances promoted when single sources were highly confident, as well as a number of methods pairing together to promote instances when the individual methods were less confident.

In total, MEC learned 531 coupled functions, since 3 different subsystems (CMC, CPL, and CSEAL) learn about 123 categories, and 3 different subsystems (CPL, CSEAL, and RL) learn about 55 relations.

In the first 22 iterations, MEC learned 88,502 facts with an estimated precision of 90%, and in 66 iterations, MEC learned over 242,000 facts with an estimated precision of 74%. In previous chapters, MBL learned 9,463 facts with an estimated precision of 92%, CPL learned 18,310 facts with an estimated precision of 81%, and CSEAL learned 12,522 facts with an estimated precision of 86%. Thus, at similar levels of precision, MEC learned far more facts than previous methods.[5]

The importance of our design principle of using components which make mostly independent errors is generally supported by the results. More than half of the beliefs were promoted based

[4]`http://rtw.ml.cmu.edu/acarlson_thesis/`

[5]To be fair, it should be noted that MEC started with an ontology that was roughly twice as large as the one used by MBL, CPL, and CSEAL.

on evidence from multiple sources. However, in looking at errors made by the system, it is clear that CPL and CMC are not perfectly uncorrelated in their errors. As an example, for the category bakedGood, CPL learns the pattern "X are enabled in" because of the believed instance "cookies." This leads CPL to extract "persistent cookies" as a candidate bakedGood. CMC outputs high probability for phrases that end in "cookies," and so "persistent cookies" is promoted as a believed instance of bakedGood.

This behavior suggests an opportunity for leveraging more human interaction in the learning process. Currently, such interaction is limited to approving or rejecting inference rules proposed by RL. However, we plan to explore other forms of human supervision, limited to approximately 10–15 minutes per day. In particular, *active learning* [Settles, 2009] holds much promise by allowing MEC to ask "queries" about its beliefs, theories, or even features about which it is uncertain. For example, a pattern like "X are enabled in" is only likely to occur with a few instances of the bakedGood category. This could be a poor pattern that leads to semantic drift, or it could be an opportunity to discover some uncovered subset of the bakedGood category. If MEC can adequately identify such opportunities for knowledge, a human can easily provide a label for this single pattern and convey a substantial amount of information in just seconds. Previous work has shown that labeling features (e.g., context patterns) rather than instances can lead to significant improvements in terms of reducing human annotation time [Druck *et al.*, 2009].

5.6 Conclusion

We have proposed an architecture for coupling the learning of many extraction techniques, and described a partial implementation of that architecture which uses four subsystem components that learn to extract knowledge in complimentary ways. After running for 67 days, this implementation populated a knowledge base with over 242,000 facts with an estimated precision of 74%. Analysis of the results shows that each of the four extraction methods contributes positively to these results.

These results confirm that our coupled semi-supervised learning approaches can scale to hundreds of predicates and can benefit from using a diverse set of extraction methods.

Chapter 6

Coupled Semi-Supervised Logistic Regression

Abstract

In this chapter, we consider how to couple the semi-supervised learning of a different kind of model from those considered earlier in the thesis: logistic regression models. Specifically, we consider learning many binary logistic regression classifiers when many (but not all) pairs of classes are known to be mutually exclusive. We present a method that uses unlabeled data through a penalty function that regularizes the training of classifiers by penalizing violations of mutual exclusion constraints. We then apply this idea to training classifiers which decide if a noun phrase is a member of some specific category. Semi-supervised training of such classifiers is shown to improve performance relative to supervised-only training , and to slightly improve performance relative to the Coupled Pattern Learner method presented in Chapter 3. We speculate that use of these and other penalty functions could provide an alternative to the methods for coupled semi-supervised learning presented in previous chapters, with the advantage that the models being learned are principled, probabilistic models that are easy to train and can be applied to any noun phrase.

6.1 Introduction

In this chapter, we consider how to couple the semi-supervised training of logistic regression models. Specifically, we consider training many binary logistic regression classifiers when many (but not all) pairs of classes being learned are known to be mutually exclusive. We present a method that uses unlabeled data through a penalty function that is added to the typical L_2-regularized logistic regression objective function. This penalty function regularizes the training of the classifiers by penalizing violations of mutual exclusion constraints in the predictions of those classifiers on unlabeled examples. This yields an objective function where the optimization of weights trades off fitting the labeled data well, having weights with small magnitude, and avoiding labeling unlabeled examples with non-zero posterior probabilities for pairs of mutually exclusive classes.

In an experimental evaluation, this general method is applied to training classifiers which decide if a noun phrase is a member of some specific category, given features that describe how strongly that noun phrase co-occurs with hundreds of thousands of different contextual patterns in a large web corpus. Semi-supervised training of such classifiers is shown to improve performance relative to supervised-only training, and to slightly improve performance relative to the Coupled Pattern Learner method presented in Chapter 3.

We speculate that use of these and other penalty functions to couple the learning of logistic regression models could provide an alternative to CPL, the method for coupled semi-supervised learning presented in Chapter 3, with several advantages:

- The models being learned are principled, probabilistic models.
- The models can be trained easily using gradient-based optimization methods.
- The models easily leverage information about the strength of co-occurrence between a noun phrase and a contextual pattern (compared to the binary features used by previous models in the thesis).

- The models offer higher recall because they can make predictions about any noun phrase.
- The models can easily incorporate new types of features (e.g., capitalization of noun phrases, prefixes and suffixes of noun phrases, distributional similarity with seeds).

6.2 Approach

Our approach to coupling the semi-supervised training of many logistic regression models adds a penalty function to the standard L_2-regularized logistic regression objective function. In explaining our approach, we start with a review of supervised binary logistic regression, and then explain our penalty function and training method.

6.2.1 Supervised Logistic Regression

Logistic regression is a popular method for discriminative probabilistic modeling. For a binary classification task, we model the probability of an instance x from instance space $\mathcal{X} = \mathbb{R}^D$ being a positive example of a class y (where $y \in \mathcal{Y} = \{0, 1\}$) as:

$$p_\theta(y = 1|x) = \frac{\exp\left(\sum_i \theta_i x_i\right)}{1 + \exp\left(\sum_i \theta_i x_i\right)}$$

where θ is a parameter vector in $\mathbb{R}^D$ (the same dimensionality as the instance x), and θ_i and x_i refer to the ith value in the vectors θ and x, respectively[1].

Given a collection L of labeled training examples that consist of (x, y) pairs, where x is a training instance, and y is the label for that instance, θ is learned by maximizing the conditional log-likelihood of the labels of the training data (thus obtaining a maximum conditional likelihood estimate of θ):

[1]Many sources in the literature negate the product between θ and x; this has no effect on the results.

$$l(\theta, L) = \sum_{(x,y)\in L} \log p_\theta(y|x)$$

In practice, we minimize an objective function that is the negative conditional log-likelihood:

$$Obj(\theta, L) = - \sum_{(x,y)\in L} \log p_\theta(y|x)$$

To combat overfitting, often the objective function contains the conditional log-likelihood supplemented with a regularization penalty. In fact, in problems where the dimensionality of the instance space exceeds the number of training examples, regularization is necessary for the problem to be well-posed. We add an L_2 regularization penalty: (this is equivalent to a Gaussian prior over the weights with mean 0):

$$Obj(\theta, L) = \frac{\lambda_R}{2} \sum_i \theta_i^2 - \sum_{(x,y)\in L} \log p_\theta(y|x)$$

where λ_R is a parameter which trades off how well the model fits the training data with the preference for small weights. Minimizing this objective will yield a MAP estimate of θ.

This objective function can be minimized using gradient descent-based methods, where the gradient with respect to weight θ_i is:

$$\frac{\partial Obj(\theta, L)}{\partial \theta_i} = \lambda_R \theta_i - \sum_{(x,y)\in L} x_i \left[y - p_\theta(y = 1|x)\right]$$

Our objective function is convex, and so our gradient descent procedure will converge to the global minimum.

This gradient can be interpreted intuitively. Training examples where the label is perfectly predicted (i.e., $p(y|x) = 1$ or $p(y|x) = 0$) do not contribute to the gradient, while training examples where the label is not perfectly predicted contribute to the gradient in proportion with the error in the prediction and the strength of each feature in that example. The regularization term for each weight contributes to the gradient proportionally to the absolute value of that weight.

Learning Many Models Simultaneously

Consider the setting where we are learning K binary classifiers over the same instance space $\mathcal{X}$. We have a different set of training data for each classifier, and we denote the set of labeled data for the kth classifier by L_k.

Let θ be a $K \times D$ parameter matrix, where θ_k is the parameter vector for the kth classifier, and $\theta_{k,i}$ is the ith entry in the parameter vector for the kth classifier.

We can learn parameters for all K classifiers simultaneously by summing their objective functions to yield:

$$Obj(\theta, L) = \sum_k \left(\left[\frac{\lambda_R}{2} \sum_i \theta_{k,i}^2 \right] - \sum_{(x,y) \in L_k} \log p_{\theta_k}(y|x) \right) \tag{6.1}$$

The sum of convex functions is convex, so this function is convex, and optimization with gradient descent will find the global minimum.

The partial derivative with respect to parameter $\theta_{k,i}$ is only affected by the training data for the kth classifier:

$$\frac{\partial Obj(\theta, L)}{\partial \theta_{k,i}} = \lambda_R \theta_{k,i} - \sum_{(x,y) \in L_k} x_i \left[y - p_{\theta_k}(y_k = 1|x) \right] \tag{6.2}$$

6.2.2 Coupled Semi-Supervised Logistic Regression

Consider the setting from the previous section where we are simultaneously learning parameters for K binary classifiers. Assume that we have domain knowledge which states that some pairs of classes among the K classes are mutually exclusive. Formally, we have some set of unordered pairs $M \subseteq K \times K$ where each pair specifies the indices of a pair of mutually exclusive classes. For example, if the first and second classes are mutually exclusive, then the pair $(1, 2)$ will be in M.

Assume that we also have a collection of unlabeled data U that consists only of instances x without labels. To use this unlabeled data in coupling the learning of our classifiers, our model penalizes cases where two mutually exclusive classes predict high posterior probabilities for an unlabeled instance by using a penalty function $P(\theta, U)$ that sums the product of the posteriors:

$$P(\theta, U) = \lambda_P \sum_{(a,b)\in M} \sum_{x\in U} p_{\theta_a}(y_a = 1|x) p_{\theta_b}(y_b = 1|x) \tag{6.3}$$

For the kth class, let M_k denote the set of classes that are mutually exclusive with class k. Then the partial derivative of the penalty with respect to parameter $\theta_{k,i}$ is:

$$\frac{\partial P(\theta, U)}{\partial \theta_{k,i}} = \lambda_P \sum_{m\in M_k} \sum_{x\in U} x_i p_{\theta_k}(y_k = 1|x)(1 - p_{\theta_k}(y_k = 1|x)) p_{\theta_m}(y_m = 1|x) \tag{6.4}$$

where λ_P is a parameter that controls the strength of $P(\theta, U)$ in the overall objective function.

This gradient has an intuitive interpretation. For some instance x, if classes a and b are mutually exclusive and predict non-zero posterior probability for x, then the gradient will be negative for all positive features in the instance (and positive for all negative features). This puts pressure on the classifiers for both classes to decrease the predicted posterior for x. The gradient for a becomes more negative as the degree of uncertainty of the label for class a increases (i.e., gets closer to 0.5), and also as the predicted posterior for class b increases. Whichever class has the lower predicted posterior is pressured to reduce its predicted posterior more than the other class.

The objective function is obtained by adding the mutual exclusion penalty (Equation 6.3) to the supervised objective function (Equation 6.1), and the gradient of that objective function by adding the gradient of the mutual exclusion penalty (Equation 6.4) to the standard supervised gradient (Equation 6.2). The constants λ_R and λ_P balance the contributions of the likelihood, regularization, and penalty term in the objective function.

The new objective is not convex, because the penalty (Equation 6.3) is not convex. To see this, consider one parameter $\theta_{k,i}$ in the objective (assume that class k is mutually exclusive with at least one other class). If all other parameters are held fixed, and this parameter is varied, the resulting function is a scaled and shifted sigmoid function, which is clearly not convex (this is obvious from a plot of the sigmoid function).

Thresholding the Penalty

In practice, we found that it is desirable to only penalize cases where two classifiers for mutually exclusive classes predict posterior probabilities greater than some threshold. Otherwise, weights are driven down by pressure to predict a posterior of 0 on unlabeled examples, when in practice we are most interested in penalizing larger values of the posteriors.

First, let $U^{\tau}_{a,b}$ denote the subset of the unlabeled data U where the predicted posterior probabilities for classes a and b both exceed τ (we use $\tau = 0.2$ in the experiments that follow):

$$U^{\tau}_{a,b} = \{x \in U | p_{\theta_a}(y_a = 1|x) > \tau \wedge p_{\theta_b}(y_b = 1|x) > \tau\}$$

We define a thresholded penalty P_τ which ignores unlabeled examples where the posterior probabilities do not exceed τ by summing over a subset of U:

(6.5)
$$P_\tau(\theta, U) = \lambda_P \sum_{(a,b)\in M} \sum_{x \in U^{\tau}_{a,b}} p_{\theta_a}(y_a = 1|x) p_{\theta_b}(y_b = 1|x)$$

The partial derivative of this penalty with respect to $\theta_{i,k}$ is similar to Equation 6.4, with the difference being that it sums over a subset of the unlabeled data U rather than all of it:

$$\frac{\partial P_\tau(\theta, U)}{\partial \theta_{k,i}} = \lambda_P \sum_{m \in M_k} \sum_{x \in U^\tau_{k,m}} x_i p_{\theta_k}(y_k = 1|x)(1 - p_{\theta_k}(y_k = 1|x)) p_{\theta_m}(y_m = 1|x) \tag{6.6}$$

This modified penalty term is discontinuous, which could cause issues during parameter learning. However, we did not observe problems during our experiments. This may be due to the fact that parameters are initialized using the parameters learned by supervised logistic regression, and then the mutual exclusion penalty is driven down. Since the penalty is monotonic, the gradient will indicate a good direction for reducing the penalty, in spite of the discontinuity.

6.2.3 Parameter Estimation

In our experiments, we use L-BFGS[2] [Byrd *et al.*, 1994] to estimate parameters. L-BFGS is a quasi-Newton optimization procedure which estimates the Hessian using the first derivative of the objective function. L-BFGS uses limited memory by not estimating the full Hessian, which would be infeasible due to the millions of parameters in our experiments. L-BFGS is widely used for parameter estimation in exponential models such as Conditional Random Fields [Sutton and Mccallum, 2006]. To use L-BFGS, we need only provide routines that calculate the objective and gradient given a parameter vector.

6.3 Experimental Evaluation

We conducted an experimental evaluation to answer the following questions:

- Does semi-supervised learning of logistic regression classifiers using mutual exclusion penalties to couple the learning of the classifiers improve the precision of the learned classifiers?

[2]We use the Java port of L-BFGS available at `http://riso.sourceforge.net/`

- How do the results with semi-supervised logistic regression compare with results from Coupled Pattern Learner from Chapter 3, since they provide alternative methods for semi-supervised learning to extract new instances of categories from seed instances and mutual exclusion relationships?

To answer these questions, we ran the following algorithms using the same ontology as the one used in Chapter 5:

- **LR$_{\textbf{Sup}}$**: Supervised Logistic Regression, which uses Equation 6.1 as its objective function
- **LR$_{\textbf{Semi-sup}}$**: Semi-Supervised Logistic Regression, which uses Equation 6.1 added to Equation 6.5 as its objective function
- **CPL**: Coupled Pattern Learner, as described in Chapter 3

In experiments with LR$_{\text{Sup}}$ and LR$_{\text{Semi-sup}}$, a classifier was learned for each category where the classification task was to decide if a particular noun phrase was an instance of that category. Thus, individual noun phrases corresponded to individual examples in our evaluation. Features describing a noun phrase were generated based on co-occurrence of that noun phrase with the same types of contextual patterns used by CPL in previous chapters over the same 2.5-billion sentence corpus of text used in Chapter 5. The comparison between LR$_{\text{Sup}}$ and LR$_{\text{Semi-sup}}$ used only the seeds for categories (up to 15 seed instances). The comparison between LR$_{\text{Semi-sup}}$ and CPL used some extra seed instances (CPL propagates seed instances from relations to categories, so LR$_{\text{Semi-sup}}$ was allowed to use these instances for fairness). More details are given below.

6.3.1 Training the Logistic Regression Models

Features for Noun Phrases

For features, we started with the contextual patterns recognized by CPL (which are defined by part-of-speech tag sequences), but, for efficiency reasons, used only the seed patterns plus the patterns

Context	$n(c)$	$\log_e(\lvert P\rvert/n(c))$
X determine the feasibility	83	11.69
X please e-mail	664	9.61
X use like	125	11.28
I now know X	1411	8.86
I upgraded X	766	9.47
congress in X	2568	8.26
luggage of X	1771	8.63
motor for X	2931	8.12
public transportation in X	1216	9.00
year , due to X	1009	9.19

Table 6.1: 10 randomly chosen example contextual patterns used as features in $\text{LR}_{\text{Semi-sup}}$, the number of different noun phrases that each occurs with, and idf weight for the contextual pattern.

that occur at least 2,500 times in the ClueWeb09 data set. This yielded 237,074 contextual pattern features.

Instead of using binary features that indicate whether or not a noun phrase and pattern co-occur in the ClueWeb09 data, we wanted to use real-valued features that capture how strongly a noun phrase and a pattern co-occur. During development with LR_{Sup}, we found that the obvious method of using raw co-occurrence counts yielded poor results. To learn how text categorization experts typically weight features, we looked at the pre-processing performed on the RCV1 collection by Lewis *et al.* [2004]. RCV1 is a collection of over 800,000 Reuters newswire articles released by Reuters [Rose *et al.*, 2002]. Lewis et al. benchmarked numerous text categorization methods on the data set, and described their pre-processing steps used to create feature vectors for text documents in detail. Our method draws from Section 7 of their paper, which in turn uses the Cornell *ltc* term weighting of Buckley *et al.* [1994]. The weight of a feature for contextual pattern c in the feature vector for noun phrase p is 0 if c and p do not co-occur in the ClueWeb09 data. If they do co-occur, the weight is:

$$w_p(c) = (1 + \log_e(n(c,p))) \times \log_e(|P|/n(c))$$

where $n(c,p)$ is the number of times contextual pattern c co-occurs with noun phrase p in our corpus, $n(c)$ is the number of unique noun phrases that c co-occurs with, and $|P|$ is the number of noun phrases in our data (in our experiments, 9,891,655, which is the number of noun phrases that occur at least 50 times and with at least 10 unique contextual patterns).

Examples of contextual patterns and their IDF weights are shown in Table 6.1.

In addition to the 237,074 contextual pattern features, we used three additional features:

- A "default" feature, whose value is 1.0 for all noun phrases
- A "capitalized" feature, whose value is 1.0 if the noun phrase starts with a capital letter, and 0.0 otherwise
- An "uncapitalized" feature, whose value is 1.0 minus the value of the "capitalized" feature

Following common practice in text categorization research (including the methodology of Lewis *et al.* [2004]), we divide the elements of each feature vector by the Euclidean norm of the vector, so that each feature vector has unit length.

The regularization coefficients λ_R for the three non-contextual pattern features are divided by 10, so that the models can more easily learn the class priors and capitalization preferences (otherwise, these features can be drowned out by the hundreds of thousands of contextual pattern features).

Training LR$_{\text{Sup}}$

Positive examples for training LR$_{\text{Sup}}$ were obtained using the seeds for each category. Negative examples were obtained by using the seeds for mutually exclusive categories, so that "Country" seeds became negative examples for the "City" category, as in previous chapters.

To train LR_{Sup}, we initialize all weights to 0, and run L-BFGS for up to 100 iterations. We found that the default tolerance parameters were too coarse, which led to L-BFGS terminating before it converged. Using a tolerance setting of 10^{-10}, which controls how small the magnitude of the gradient must be to terminate the optimization procedure early, prevented L-BFGS from terminating too early. To select λ_R, we trained models for $\lambda_R \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$, and compared the predictions for the "Body Part" and "Writer" categories. 10^{-6} was a clear winner in both cases, so we used that value in our final experiments.

Figure 6.1 shows the values of each component of the objective function after each iteration of L-BFGS. Iterations might include line search steps which are retracted, so the curve does not descend smoothly, but the optimization procedure appears to converge well. Training LR_{Sup} took 20 minutes.

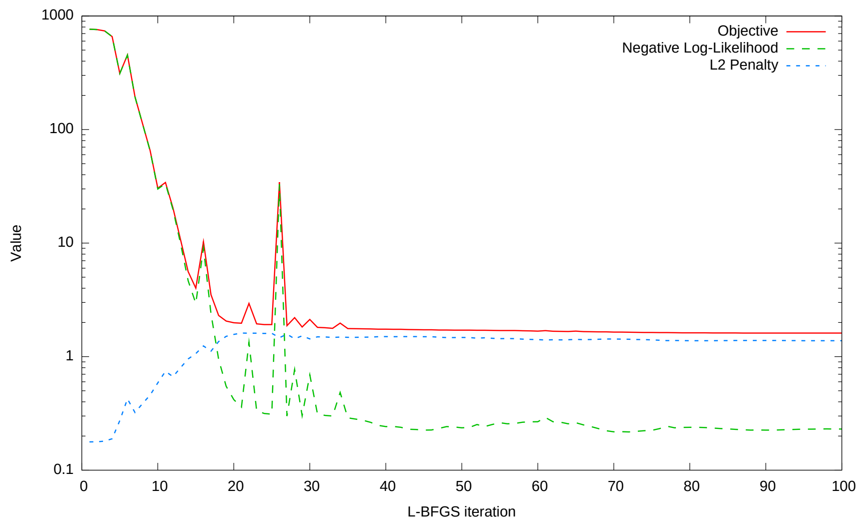


Figure 6.1: The LR_{Sup} objective function and its components versus iteration of L-BFGS. Spikes occur due to the line search performed by L-BFGS. Since weights are initialized to 0, negative log-likelihood decreases while the L2 penalty increases as the weights deviate from 0 to fit the training data.

Training $\text{LR}_{\text{Semi-Sup}}$

To train $\text{LR}_{\text{Semi-sup}}$, we use the same value of λ_R as in the LR_{Sup} experiments, and use the learned weights for LR_{Sup} as the starting point, since the objective for $\text{LR}_{\text{Semi-sup}}$ is not convex. The objective function to minimize for $\text{LR}_{\text{Semi-sup}}$ is the LR_{Sup} objective (Equation 6.1) plus the thresholded penalty function (Equation 6.5). The unlabeled noun phrases used in the penalty function were chosen by taking the noun phrases that occurred at least 5,000 times in the ClueWeb09 data, yielding 177,181 noun phrases. During development of $\text{LR}_{\text{Semi-sup}}$, we ran it on a small collection of categories from the ontology in Chapter 3, and found that a λ_P value of 0.0001 and a threshold $\tau = 0.2$ worked well. We used these values in the final experiments.

We set up L-BFGS to learn weights for up to 50 iterations. Figure 6.2 shows the values of each component of the objective function for $\text{LR}_{\text{Semi-sup}}$ after each iteration of L-BFGS. The mutual exclusion penalty starts out large relative to the other components in the objective function, and is driven down steadily as optimization proceeds. Training $\text{LR}_{\text{Semi-sup}}$ took 167 minutes.

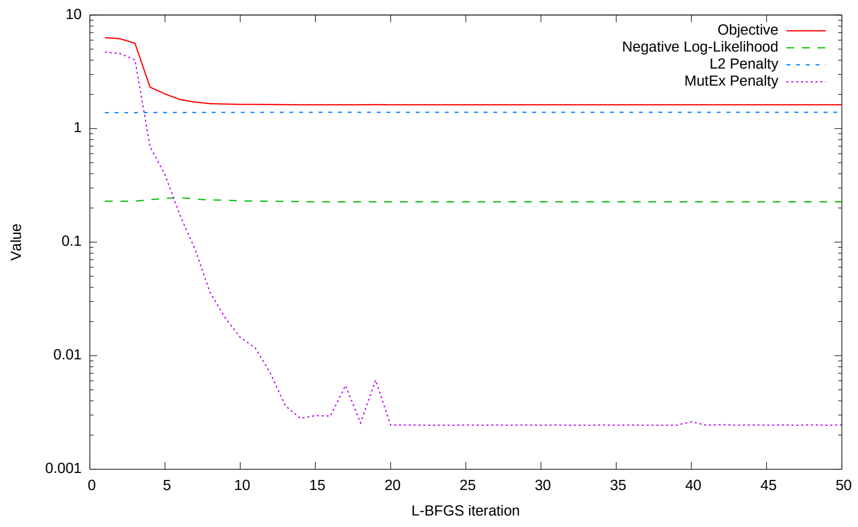


Figure 6.2: The $\text{LR}_{\text{Semi-sup}}$ objective function and its components versus iteration of L-BFGS, after initializing the weights by running LR_{Sup}. The negative log-likelihood and L_2 penalty do not change much as the mutual exclusion penalty decreases.

6.3.2 Training CPL

CPL was run for 40 iterations, and allowed to promote up to 250 instances per category per iteration. This run took 100 hours, and 61,495 category instances were promoted in total.

6.3.3 Methodology for Comparing Different Methods

We compare two different pairs of methods in our evaluation: LR_{Sup} and $\text{LR}_{\text{Semi-sup}}$, and CPL and $\text{LR}_{\text{Semi-sup}}$. In general, we use a methodology designed to detect differences between a pair of methods.

For each method and for each category, we obtain a ranked list of noun phrases. For LR_{Sup} and $\text{LR}_{\text{Semi-sup}}$, this is done by classifying all of the noun phrases that occur at least 50 times and with at least 10 unique context patterns in the ClueWeb09 data (of which there are 9,891,655), and then ranking those noun phrases by their predicted posterior probabilities of being instances of the category. Noun phrases that were seed instances for any category are removed from all ranked lists. For CPL, a ranked list for each category is obtained by sorting all promoted noun phrases by the iteration on which they were promoted, and breaking ties based on the number of promoted contextual patterns that extracted that noun phrase.

The comparisons in this evaluation were designed to detect changes in the precision of the top k noun phrases in these ranked lists for specific values of k. Generally, to compare the top k noun phrases for some category for method A and method B, we consider the sets of noun phrases in the top k noun phrases for the methods. Noun phrases which are in both sets are ignored. The precision of the noun phrases unique to set A is estimated by labeling all of those noun phrases if there are 25 or fewer, or 25 if there are more than 25, and then using the number of correct noun phrases divided by the number considered as the estimated precision. The precision of the noun phrases unique to set B is similarly estimated. Then the change in precision from method A to method B is estimated as:

$$\frac{(p_B - p_A)\,(k - |A \cap B|)}{k}$$

where p_A is the estimated precision of noun phrases unique to set A, and p_B is the estimated precision of noun phrases unique to set B.

10 categories were randomly selected for the evaluation:

- Chef
- Country
- Currency
- Economic Sector
- Fruit
- Hotel
- Lake
- Professional Organization
- River
- Visual Artist

When judging the correctness of an instance, small typographical errors were acceptable. Plurality was ignored, as was case. Adjectives were acceptable if they referred to a subcategory of the target category (e.g., “luxury hotels” as an instance of the category “Hotel”) but not if they were overly generic (e.g., “good hotel”). Counts of correct instances were not judged as correct (e.g., “3 apples” for the category “Fruit”). Labeling was not performed blind to the method producing the ranked lists. All judgments performed in the process of the evaluation are available online in the supplementary online materials for the thesis. Labeling was performed by the author.

6.3.4 Results and Discussion

Results for LR_{Sup} and $\mathrm{LR}_{\text{Semi-sup}}$

Figure 6.3 shows estimated changes in precision for ten categories and six values of k that result from moving from LR_{Sup} to $\mathrm{LR}_{\text{Semi-sup}}$. Across all categories and values of k, only three changes were estimated to be negative, so $\mathrm{LR}_{\text{Semi-sup}}$ rarely performs worse than LR_{Sup}. The results for $\mathrm{LR}_{\text{Semi-sup}}$ frequently show some gain in precision relative to LR_{Sup}, with some large gains for categories like “Country,” “Currency,” and “Economic Sector.”

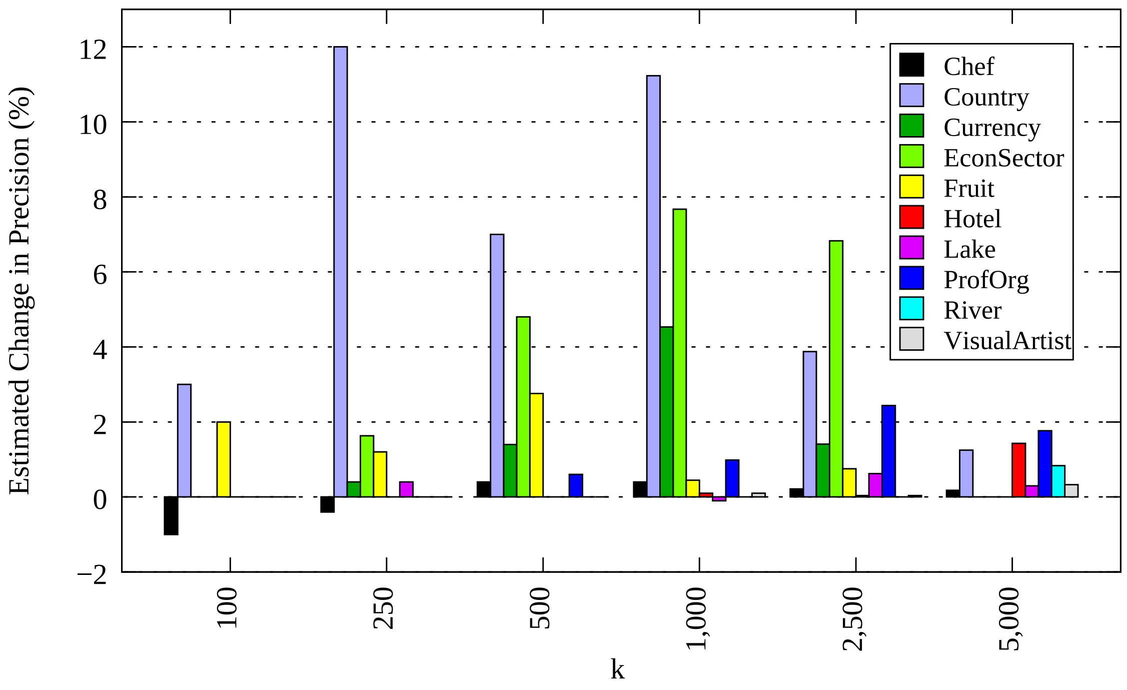


Figure 6.3: Estimates of the absolute change in precision for the top k noun phrases ranked by their posterior probabilities when moving from LR_{Sup} to $\mathrm{LR}_{\text{Semi-sup}}$ for ten different categories.

Table 6.2 shows, for each value of k, how many categories had higher estimated precision for $\mathrm{LR}_{\text{Semi-sup}}$, and how many categories had higher estimated precision for LR_{Sup}. The general trend of $\mathrm{LR}_{\text{Semi-sup}}$ improving precision relative to LR_{Sup} is clear. According to sign tests, the differences for k equal to 500, 1,000, 2,500, and 5,000 are statistically significant at the 5% level.

k	# LR$_{\text{Sup}}$ better	# LR$_{\text{Semi-sup}}$ better	Sign Test p-value
100	2	1	n/a
250	5	1	0.219
500	6	0	$0.0313_{\dagger}$
1,000	8	1	$0.0391_{\dagger}$
2,500	9	0	$0.00391_{\dagger}$
5,000	7	0	$0.0156_{\dagger}$

Table 6.2: Results of sign tests comparing the accuracy of the lists produced by LR$_{\text{Sup}}$ and LR$_{\text{Semi-sup}}$ for ten categories at various thresholds for the number of promotions. Statistically significant p-values are marked by †. LR$_{\text{Semi-sup}}$ is significantly better than LR$_{\text{Sup}}$ when considering the first 500, 1,000, 2,500, and 5,000 noun phrases for each category.

Figure 6.4 shows estimates of the precision of the noun phrases selected by LR$_{\text{Semi-sup}}$ when considering the top k noun phrases for a category. Precision was estimated by sampling 25 noun phrases uniformly at random from the top k noun phrases. As expected, as k increases, precision tends to drop. Many categories perform quite well; “Economic Sector,” “Hotel”, “Professional Organization,” “River,” and “Visual Artist” have quite good precisions for the top 2,500 and 5,000 noun phrases.

To examine the effect of semi-supervised training on feature weights relative to supervised training, we looked at the contextual patterns which had their weights change the most between the LR$_{\text{Sup}}$ run and the LR$_{\text{Semi-sup}}$ run for the “Currency” category. Illustrative examples were picked from these patterns and are shown in Table 6.3. The top five patterns shown had their weights decrease significantly, while the bottom five patterns shown had their weights increase significantly. The patterns for which weights decrease the most are sensible, because they are overly general: all of them co-occur frequently with noun phrases that are not currencies (e.g., “time” and “energy” with “X spent with, and “donations” with “X were collected at”). The patterns for which the weights increase the most are also sensible, because they seem like reasonable high-precision patterns for the “Currency” category.

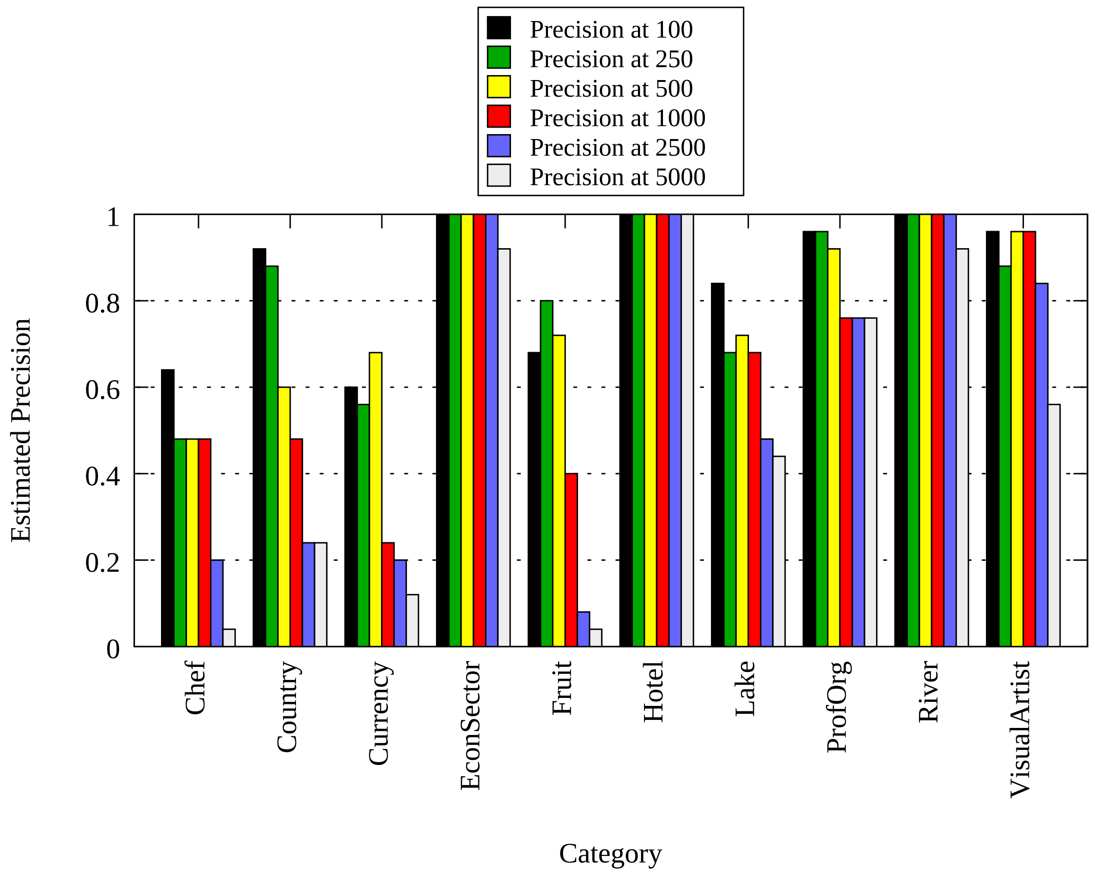


Figure 6.4: Estimates of the precision of $\mathrm{LR}_{\text{Semi-sup}}$ for the top k noun phrases ranked by their posterior probabilities, for a variety of values of k and a variety of categories.

Context	LR_{Sup} weight	$\text{LR}_{\text{Semi-sup}}$ weight	Change in weight
X are accepted for	0.81	0.28	-0.53
X were spent in	1.47	1.00	-0.47
X spent with	0.15	-0.29	-0.43
X were collected at	0.44	0.11	-0.33
X are allowed for	0.47	0.15	-0.32
worth thousands of X	4.18	4.34	0.15
many billions of X	3.16	3.29	0.13
X amount of	1.55	1.65	0.10
hard earned X	3.45	3.55	0.10
prices are in X	4.07	4.16	0.09

Table 6.3: Selected contextual patterns for which the weights for the “Currency” classifier decreased (top) and increased (bottom) significantly between supervised and semi-supervised training.

Similarly, to see the effect of semi-supervised training on predictions of posterior probabilities, we considered the noun phrase “More time” which incorrectly had a high predicted posterior probability for the “Currency” category according to LR_{Sup}, but had a low predicated posterior according to $\text{LR}_{\text{Semi-sup}}$. Table 6.4 shows the posterior probabilities predicted by LR_{Sup} for the noun phrase “More time” that exceeded 0.2, as well as the posterior probabilities predicted for the same noun phrase and the same categories by $\text{LR}_{\text{Semi-sup}}$. The predicted posterior probabilities decrease as we would expect, and the category with highest posterior predicted by $\text{LR}_{\text{Semi-sup}}$, “Abstract Thing,” makes sense since “More time” is indeed an abstract thing[3].

Broadly, $\text{LR}_{\text{Semi-sup}}$ clearly outperforms LR_{Sup} in these experiments. This provides support for the conclusion that semi-supervised training of logistic regression classifiers using mutual exclusion penalties results in more accurate classifiers than supervised training. Additionally, the changes that occurred in the feature weights for the “Currency” classifier after adding the mu-

[3]“Abstract Thing” is a non-leaf node in our ontology of categories, with subcategories like “Emotion” and “Academic Field”

Category	LR_{Sup} Posterior	$\text{LR}_{\text{Semi-sup}}$ posterior
Abstract Thing	0.90	0.55
Agent	0.42	0.23
Company	0.44	0.18
Currency	0.74	0.15
Magazine	0.68	0.19
Media Company	0.58	0.19
Organization	0.43	0.30
Publication	0.53	0.17

Table 6.4: Posterior probabilities for certain categories for the noun phrase "More time" predicted by the supervised model and semi-supervised model.

tual exclusion penalty matched intuitive expectations; $\text{LR}_{\text{Semi-sup}}$ appears to prefer more specific contextual patterns than LR_{Sup}.

Another question answered by this evaluation was: how do logistic regression methods perform on this task, generally? It appears that they work well, and merit further consideration in future work. The categories with high precision for the top 5,000 noun phrases (e.g., "River," "Economic Sector," and "Hotel") provide particularly compelling evidence for this point.

Results comparing CPL and $\text{LR}_{\text{Semi-sup}}$

In comparing CPL and $\text{LR}_{\text{Semi-sup}}$, we wanted to compare the precision of the two methods at low and high levels of recall. To accomplish this, we compared the precision of the top k noun phrases for the same ten categories as above, for $k = 100$ and also for k equal to the number of noun phrases promoted by CPL for that category after 40 iterations. These counts are shown in Table 6.5.

Changes in precision were estimated for these values of k using the methodology described in Section 6.3.3. The estimated changes in precision from CPL to $\text{LR}_{\text{Semi-sup}}$ are shown in Figure 6.5. When comparing the top 100 noun phrases for each category, CPL performs better for 4 categories, while $\text{LR}_{\text{Semi-sup}}$ performs better for 5. When comparing the top noun phrases using the number

Category	Number of CPL Promotions
Chef	134
Country	414
Currency	214
EconSector	350
Fruit	297
Hotel	418
Lake	1456
ProfOrg	1357
River	573
VisualArtist	809

Table 6.5: Numbers of instances promoted by CPL in the 40 iteration run used in the comparison with $\text{LR}_{\text{Semi-sup}}$. The precisions of these promotions was compared with $\text{LR}_{\text{Semi-sup}}$.

of noun phrases promoted by CPL for each predicate, CPL performs better for 3 categories, while $\text{LR}_{\text{Semi-sup}}$ performs better for 7 categories. This suggests that $\text{LR}_{\text{Semi-sup}}$ may have higher precision than CPL at high levels of recall, but the difference is not significance according to a Sign Test.

An item worth noting is that CPL has an advantage due to its use of seed patterns. In the first iteration of execution, CPL promotes noun phrases that occur with at least two seed patterns for a category, because they are automatically treated as promoted patterns. $\text{LR}_{\text{Semi-sup}}$, on the other hand, only includes the seed patterns as features, and has a chance to learn high weights for them. The fact that $\text{LR}_{\text{Semi-sup}}$ was competitive with CPL when considering the top 100 noun phrases for each category is more impressive when taking this difference into account.

When compared to $\text{LR}_{\text{Semi-sup}}$, CPL appears to have a preference for more frequent noun phrases. For example, the average Google estimated hit count for 10 randomly selected noun phrases in the first 100 promoted by CPL for the "Visual Artist" category was 1,013,110, while the average hit count for 10 noun phrases in the top 100 for $\text{LR}_{\text{Semi-sup}}$ was 106,940, an order of magnitude lower. This difference makes sense, because noun phrases that occur many times are likely to co-occur with at least two patterns promoted by CPL. $\text{LR}_{\text{Semi-sup}}$ normalizes all feature

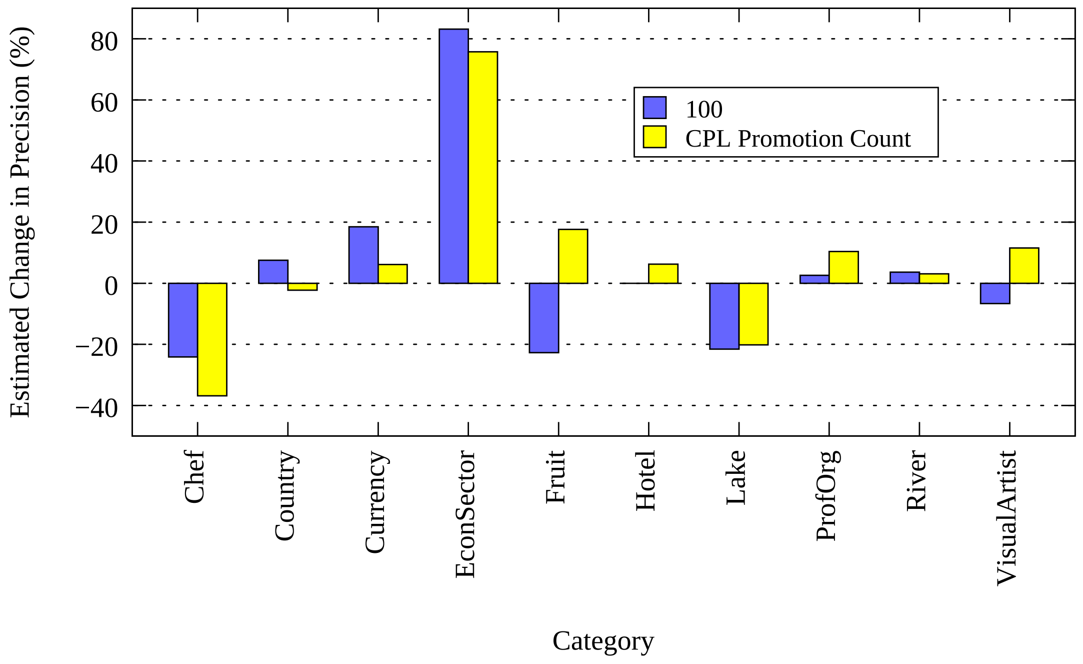


Figure 6.5: Estimates of the absolute change in precision for the top 100 noun phrases ranked by their posterior probabilities when moving from CPL to $\text{LR}_{\text{Semi-sup}}$ for ten different categories. Positive estimated changes indicate that $\text{LR}_{\text{Semi-sup}}$ had estimated precision superior to CPL.

vectors to unit length, so it has no preference for frequent noun phrases.

Generally, it appears that $\text{LR}_{\text{Semi-sup}}$ and CPL are competitive with each other. $\text{LR}_{\text{Semi-sup}}$ may be superior to CPL in precision at high levels of recall, but our evaluation was not conclusive on this point.

Supplementary Online Materials

Several types of supplementary materials from our evaluation are posted online[4], including:

- The categories used and their seed instances (listed under that AAAI 2010 supplementary online materials, because the same ontology was used)
- All weights learned by LR_{Sup} and $\text{LR}_{\text{Semi-sup}}$
- All predictions made by LR_{Sup} and $\text{LR}_{\text{Semi-sup}}$ with posteriors exceeding 0.05
- All noun phrases promoted by CPL in the 40 iteration run used in the comparison with $\text{LR}_{\text{Semi-sup}}$
- All noun phrases sampled for the evaluation, with their manually assigned labels

6.4 Conclusion

We have described a novel method for coupling the semi-supervised learning of binary logistic regression classifiers when some of the classes being learned are known to mutually exclusive. Our method is simple to implement and can be optimized using gradient-based methods. Experimental results demonstrate that our semi-supervised method, $\text{LR}_{\text{Semi-sup}}$, is more precise than the standard supervised method, LR_{Sup}, at several equivalent levels of recall. Additionally, results show that $\text{LR}_{\text{Semi-sup}}$ is competitive with CPL, and may have higher precision at high levels of recall.

[4]`http://rtw.ml.cmu.edu/acarlson_thesis/`

We consider these results encouraging. An obvious future direction is to add classifiers which decide if two noun phrases are an instance of a binary relation, and to couple those classifiers with the category classifiers using type-checking constraints.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have seen that coupled semi-supervised learning methods can improve the learning of textual patterns in Chapter 3 and web page wrapper inducers in Chapter 4. The high precision of facts that are independently extracted by both methods was also demonstrated in Chapter 4. Chapter 5 showed that we can scale to hundreds of predicates and several extraction methods, and extract nearly 250,000 facts with 74% precision. Chapter 6 shows a way to couple the semi-supervised learning of logistic regression models, offering an alternative to the bootstrap learning methods of the previous chapters, as well as a high-recall probablistic model. Additionally, coupled logistic regression appears to have equivalent, and possibly superior, precision.

Taking a broader perspective, the work in Chapters 3, 4, and 5 demonstrates that constraining bootstrap learning through coupling shows significant promise as a way to achieve high-precision semi-supervised learning. Our bootstrap learning methods are similar to Hard EM [Dempster *et al.*, 1977], where in each iteration we commit to assignments for latent variables where we are most confident, and then retrain models based on all assigned variables.

The success of these methods also provides support for the usefulness of the Constraint-Driven Learning algorithm [Chang *et al.*, 2007]. Their work has the same general spirit of using constraints to limit the ways that unlabeled data can be labeled in a semi-supervised learning setting. In their algorithm, though, the top K ways to segment a piece of text are added as labeled examples (rather than just the highest-ranked way), and examples are relabeled at each iteration (rather than being labeled only once).

The work in Chapter 6 demonstrates that simultaneous learning of many logistic regression models can be improved using penalties that couple the learning of those functions. This work suggests that further study of the use of penalty functions to couple the learning of exponential models could be worthwhile.

7.2 Future Work

7.2.1 General Coupled Semi-Supervised Learning

General Study of Coupled Logistic Regression

In Chapter 6, Coupled Logistic Regression was shown to improve the precision of classifiers for extraction of category instances. These results suggest that it could be worthwhile to apply CLR to other problems where many classes are being learned and some are known to be mutually exclusive (e.g., text classification, object recognition).

Theoretical Analysis of Coupled Semi-Supervised Learning

Study of the theoretical properties of coupled semi-supervised learning deserves consideration. Questions that could be pursued include: How can we theoretically characterize the value of a coupling constrant? How does sample complexity vary with the number of functions being learned and the types and numbers of coupling constraints that are known about them?

7.2.2 Future Work within our Case Study

Entity Resolution

In this thesis, we learn facts about strings. No connection is made between similar strings that refer to the same real-world entities. For example, "Carnegie Mellon," "Carnegie Mellon University," and "CMU" are all treated separately. Adding the task of entity resolution to our case study could provide a new type of function to be learned that could be coupled with the category and relation extraction tasks. For a possible starting point, a web-scale approach to entity resolution which decides which pairs of strings refer to the same objects was presented by Yates and Etzioni [2009].

Relation Extraction with Coupled Logistic Regression

The Coupled Logistic Regression work in Chapter 6 learns classifiers that decide if a noun phrase is an instance of specific categories. A clear next step is to add classifiers that decide if a pair of noun phrases satisfy the relations learned in other chapters, and to couple the relation classifiers which the category classifiers through type-checking constraints.

Run for a Very Long Time

The results in Section 5 in particular suggest the question: what would happen if we ran MEC for months or years, especially with a human being in the loop to help keep the system on track? While some semantic drift was evident after 66 iterations, the errors made by the system (many of which are described in Appendix B) tended to have regularities that could perhaps be corrected with just a few labels from a human judge (e.g., Airports promoted as Cities, Catalogs promoted as Magazines). We suspect that with a human in the loop for just 15 minutes a day, millions of facts could be learned at very high precision with minimal supervision.

Appendix A

Ontology Used in Chapters 5 and 6

This appendix lists the categories and relations in the ontology used in Chapter 5 and Chapter 6. The full ontology is available online in spreadsheet form at `http://rtw.ml.cmu.edu/acarlson_thesis`. For reference while reading the thesis, we include some information about each category and relation in the ontology.

Category	Parent	Populate?	Example Seed
abstractThing	everyPromotedThing	no	
academicField	abstractThing	yes	Astronomy
actor	person	yes	Catherine Zeta-Jones
agent	everyPromotedThing	no	
amphibian	vertebrate	yes	frogs
animal	agent	yes	ants
arthropod	invertebrate	yes	ants
athlete	person	yes	Alex Rodriguez
awardTrophyTournament	abstractThing	yes	Stanley Cup
bakedGood	food	yes	muffins
beverage	food	yes	coffee
bird	vertebrate	yes	robins
blog	website	yes	Huffington Post

Table A.1: Categories in the ontology used in Chapters 5 and 6 (continued on the next page)

Category	Parent	Populate?	Example Seed
boardGame	abstractThing	yes	backgammon
bodyPart	item	yes	arm
book	abstractThing	yes	Freakonomics
building	location	no	barns
buildingFeature	item	yes	arches
buildingMaterial	abstractThing	yes	concrete
candy	food	yes	Skittles
cardGame	abstractThing	yes	Poker
celebrity	person	yes	Tom Cruise
cellType	abstractThing	yes	red blood cells
ceo	person	yes	Alain Belda
characterTrait	abstractThing	yes	courage
chef	person	yes	Thomas Keller
chemical	abstractThing	yes	ethylene glycol
city	location	yes	Antwerp
clothing	item	yes	coats
coach	person	yes	Bobby Bowden
cognitiveActions	abstractThing	yes	thinking
company	organization	yes	AFLAC
condiment	food	yes	ketchup
conference	abstractThing	yes	ICML
consumerElectronicItem	item	yes	video games
continent	location	no	Africa
country	location	yes	Austria
currency	abstractThing	yes	dollars
date	abstractThing	no	September 11 , 2001
dayOfWeek	date	no	Monday
disease	abstractThing	yes	lung cancer
economicSector	abstractThing	yes	advertising
election	event	no	2004 presidential election
emotion	abstractThing	yes	anxiety
ethnicGroup	abstractThing	yes	Americans
event	abstractThing	no	

Table A.1: Categories in the ontology used in Chapters 5 and 6 (continued on the next page)

Category	Parent	Populate?	Example Seed
eventOutcome	abstractThing	yes	success
everyPromotedThing		no	
female	person	yes	Madonna
fish	vertebrate	yes	sharks
food	item	no	tomatoes
fruit	food	yes	apples
fungus	item	yes	mushrooms
furniture	item	yes	beds
geometricShape	abstractThing	yes	squares
hobby	abstractThing	yes	coin collecting
hotel	building	yes	Grand Hyatt Dubai
householdItem	item	yes	bells
invertebrate	animal	yes	worms
item	everyPromotedThing	no	
journalist	person	yes	Ron Cook
kitchenItem	item	yes	bottle
lake	location	yes	Lake Michigan
landscapeFeatures	location	yes	forests
location	everyPromotedThing	no	Allegheny County
magazine	publication	yes	Readers Digest
male	person	yes	Barack Obama
mammal	vertebrate	yes	bears
mediaCompany	company	no	
mediaType	item	yes	books
medicalProcedure	abstractThing	yes	appendectomy
militaryEventType	abstractThing	yes	attacks
mlAlgorithm	abstractThing	yes	SVMs
mlArea	abstractThing	yes	supervised learning
mlAuthor	person	yes	Tom Mitchell
mlConference	conference	yes	ICML
mlDataset	abstractThing	yes	Iris
mlMetric	abstractThing	yes	precision
mlSoftware	abstractThing	yes	Minorthird

Table A.1: Categories in the ontology used in Chapters 5 and 6 (continued on the next page)

Category	Parent	Populate?	Example Seed
month	date	no	January
monument	building	yes	Eiffel Tower
mountain	location	yes	Mount Everest
mountainRange	location	yes	Rocky Mountains
movie	abstractThing	yes	The Sound of Music
museum	building	yes	Louvre
musicAlbum	abstractThing	yes	Abbey Road
musicArtist	agent	yes	Beatles
musicGenre	abstractThing	yes	rock
musician	person	yes	John Lennon
musicInstrument	item	yes	guitar
newspaper	publication	yes	Pittsburgh Post-Gazette
officeItem	item	yes	staplers
organization	agent	no	
park	location	yes	Central Park
perceptionAction	abstractThing	yes	listening
perceptionEvent	abstractThing	yes	sound
person	agent	no	
personalCareItem	item	yes	toothpaste
physicalAction	abstractThing	yes	hitting
physicalCharacteristic	abstractThing	yes	hardness
physicsTerm	abstractThing	yes	acceleration
plant	item	yes	flowers
politician	person	yes	Barack Obama
product	item	yes	Accord
profession	abstractThing	yes	cooks
professionalOrganization	organization	yes	ACM
protein	abstractThing	yes	ATP-binding cassette
publication	mediaCompany	yes	Readers Digest
radioStation	mediaCompany	yes	WBUR
recordLabel	mediaCompany	yes	Atlantic Records
religion	abstractThing	yes	Catholicism
reptile	vertebrate	yes	alligators

Table A.1: Categories in the ontology used in Chapters 5 and 6 (continued on the next page)

Category	Parent	Populate?	Example Seed
restaurant	building	yes	French Laundry
river	location	yes	Nile River
room	location	yes	kitchens
scientist	person	yes	Albert Einstein
socioPolitical	abstractThing	yes	democracy
sport	abstractThing	yes	badminton
sportsEquipment	item	yes	bats
sportsEvent	event	no	2004 Olympic Summer Games
sportsGame	sportsEvent	yes	2001 Super Bowl
sportsLeague	organization	yes	NFL
sportsTeam	organization	yes	Boston Celtics
stadiumOrEventVenue	building	yes	PNC Park
stateOrProvince	location	yes	Florida
street	location	yes	Forbes Avenue
televisionNetwork	mediaCompany	no	ABC
televisionStation	mediaCompany	yes	WHDH
tool	item	yes	chisels
university	organization	yes	California Institute of Technology
vegetable	food	yes	carrots
vehicle	item	yes	airplanes
vertebrate	animal	yes	
videoGame	product	yes	Super Mario Brothers
visualArtForm	abstractThing	yes	painting
visualArtist	person	yes	Leonardo da Vinci
visualArtMovement	abstractThing	yes	Italian Renaissance
weatherPhenomenon	abstractThing	yes	fog
website	company	yes	Google
writer	person	yes	Stephen Dubner
zipCode	location	yes	60611

Table A.1: Categories in the ontology used in Chapters 5 and 6

Relation	Domain	Range	Example Seed
acquired	company	company	(Berkshire Hathaway, NetJets)
actorStarredInMovie	actor	movie	(Jodie Foster, Silence of the Lambs)
affiliatedWith	televisionStation	televisionNetwork	(WHDH, NBC)
athleteCoach	athlete	coach	(Kobe Bryant, Phil Jackson)
athleteHomeStadium	athlete	stadiumOrEventVenue	(Mickey Mantel, Yankee Stadium)
athletePlaysForTeam	athlete	sportsTeam	(Alex Rodriguez, Yankees)
athletePlaysInLeague	athlete	sportsLeague	(Sidney Crosby, NHL)
athletePlaysSport	athlete	sport	(Alex Rodriguez, baseball)
bookWriter	book	writer	(Freakonomics, Stephen Dubner)
ceoOf	ceo	company	(Alain Belda, Alcoa)
cityLocatedInCountry	city	country	(Alexandria, Egypt)
cityLocatedInState	city	stateOrProvince	(Pittsburgh, Pennsylvania)
coachesInLeague	coach	sportsLeague	(Phil Jackson, NBA)
coachesTeam	coach	sportsTeam	(Bo Pelini, Cornhuskers)
coachWonTrophy	coach	awardTrophyTournament	(Phil Jackson, NBA Championship)
companyEconomicSector	company	economicSector	(Adobe, software)
competesWith	company	company	(AMD, Intel)
countryCurrency	country	currency	(United States, dollars)
eventDate	event	date	
hasOfficeInCity	company	city	(Alcoa, New York)
hasOfficeInCountry	company	country	(Alcoa, Iceland)
headquarteredIn	company	city	(AFLAC, Columbus)
leagueStadiums	sportsLeague	stadiumOrEventVenue	(MLB, PNC Park)
mlAreaExpert	mlArea	mlAuthor	(Neural Networks, Tom Mitchell)
mlAuthorOfSoftware	mlAuthor	mlSoftware	(William Cohen, Minorthird)
musicArtistGenre	musicArtist	musicGenre	(Yo Yo Ma, classical)

Table A.2: Relations in the ontology used in Chapters 5 (continued on the next page)

Category	Parent	Populate?	Example Seed
musicianInMusicArtist	musician	musicArtist	(John Lennon, Beatles)
musicianPlaysInstrument	musician	musicInstrument	(John Lennon, guitar)
newspaperInCity	newspaper	city	(Pittsburgh Post-Gazette, Pittsburgh)
writtenAboutInPublication	person	publication	(John Lennon, Rolling Stone)
producesProduct	company	product	(Activision, Guitar Hero)
productInstanceOf	product	item	(Civic, car)
radioStationInCity	radioStation	city	(WBUR, Boston)
sportsGameAwayTeam	sportsGame	sportsTeam	
sportsGameHomeTeam	sportsGame	sportsTeam	
sportsGameLoser	sportsGame	sportsTeam	(2001 Super Bowl, New York Giants)
sportsGameSport	sportsGame	sport	(2001 Super Bowl, football)
sportsGameTeam	sportsGame	sportsTeam	
sportsGameWinner	sportsGame	sportsTeam	(2001 Super Bowl, Baltimore Ravens)
sportUsesEquipment	sport	sportsEquipment	(baseball, bat)
sportUsesStadium	sport	stadiumOrEventVenue	(baseball, PNC Park)
stadiumLocatedInCity	stadiumOrEventVenue	city	(PNC Park, Pittsburgh)
stateHasCapital	stateOrProvince	city	(Alabama, Montgomery)
stateLocatedInCountry	stateOrProvince	country	(Pennsylvania, USA)
stationInCity	televisionStation	city	(WHDH, Boston)
teamHomeStadium	sportsTeam	stadiumOrEventVenue	(Pittsburgh Pirates, PNC Park)
teammate	athlete	athlete	(Ben Roethlisberger, Willie Parker)
teamPlaysAgainstTeam	sportsTeam	sportsTeam	(Pittsburgh Steelers, Baltimore Ravens)
teamPlaysInCity	sportsTeam	city	(Astros, Houston)
teamPlaysInLeague	sportsTeam	sportsLeague	(Yankees, MLB)
teamPlaysSport	sportsTeam	sport	(Astros, baseball)
teamWonTrophy	sportsTeam	awardTrophyTournament	(Steelers, Super Bowl)

Table A.2: Relations in the ontology used in Chapters 5 (continued on the next page)

Category	Parent	Populate?	Example Seed
visualArtistForm	visualArtist	visualArtForm	(Leonardo da Vinci, painting)
visualArtistMovement	visualArtist	visualArtMovement	(Leonardo da Vinci, Italian Renaissance)
writesFor	journalist	publication	(Ron Cook, Pittsburgh Post-Gazette)

Table A.2: Relations in the ontology used in Chapter 5

Appendix B

Descriptions of Errors Made by MEC

This appendix lists selected predicates with informal descriptions of errors made for those predicates during the run of the MEC system described in Chapter 5. The contents of this appendix are discussed in Section 5.5.2. The goal of this appendix is to list dominant trends in errors made by MEC, and the discussion does not include every error made.

B.1 Categories

- **academicField**: CMC learned that starting with the words `"resource"` and `"rights"` were strong positive features, leading to promotions like `"Resource Industry"` and `"Rights Protection"`.
- **actor**: In iteration 58, CSEAL returned many improperly formatted candidates that contained markup and punctuation, like `"{________options__option_value__}_sarah_chalke"`, which led to many incorrect promotions, but the category stayed on track after that.
- **arthropod**: CMC learned that the suffix `"beetles"` was a strong positive feature, which led to promoting lots of instances like `"what are beetles"`, `"voltswagon beetles"`,

and `"research on beetles"`.

- **awardTrophyTournament**: The string `"3-2"` was promoted in iteration 10 based on evidence from CPL, which then led to lots of similar strings like `"7-1"` and `"6-5"` being promoted.
- **bakedGood**: In iteration 23, `"cookie information"`, `"encryption technology"`, and `"tracking cookies"` were promoted, which led to lots of technology-related instances being promoted. Also, strings ending in `"recipe"` were common, like `"chocolate chip cookie bars recipe"`. Finally, CMC learned that containing the word `"cookie"` is a positive feature, so strings like `"who stole cookie monsters cookies"` were promoted.
- **beverage**: CMC learned that containing the word `"coffee"` is a positive feature, and this led to many promotions like `"coffee this morning"`, `"antique coffee"`, and `"metal coffee"`.
- **boardGame**: CMC learned that the word `"life"` is a positive feature, leading to lots of promotions like `"life catering"`, `"life end"`, and `"life band"`.
- **bodyPart**: Diverged to more general terms like `"affected regions"`, `"inner lining"`, and `"upper part"`.
- **book**: Became dominated by religious terms like `"entire old testament"` and `"glorious book"`.
- **buildingMaterial**: CMC learned the prefix `"steel"` which led many bad promotions to `"steel food"`, `"steel forks"`, and `"steel kegs"`.
- **cardGame**: Learned games that are not card games, like `"american roulette"` and `"slot machines"`, and then later learned many noisy strings like `"play casino games"` and `"free game play slot"` which appear to be related to online poker.

- **cellType**: CMC learned the suffix `"cells"` which led to dozens of promotions like `"no red blood cells"` and `"formation of memory cells"`.
- **condiment**: CMC learned the prefix `"sauce"` which led to many bad promotions like `"sauce engine"`, `"sauce teaspoon"`, and `"sauce world"`.
- **economicSector**: CMC learned the suffix `"software"` which led to many bad promotions like `"problem software"`, `"need software"`, and `"outcome software"`.
- **ethnicGroup**: Many instances like `"300,000 Palestinians"`, where a number is present at the start of the instance, were promoted, which were considered errors. Also, languages like `"Slovak language"` and `"Malay language"` were also learned.
- **female**: Many male actors were learned. Apparently actors should have been specified as seeds to the "male" category, too.
- **fruit**: `"nut recipes"` and `"fruit recipes"` were promoted in iteration 6. This led to CMC learning the suffix `"recipes"`, which then led to many bad promotions like `"strawberry recipes"`, `"avocado recipes"`, and `"apples recipes"`. After this, many recipes were learned.
- **geometricShape**: Lots of instances learned like `"8 pieces"`, `"8 wedges"`, and `"9 squares"`. Also, learned words like `"chunks"` and `"bite size cubes"`.
- **invertebrate**: CSEAL extracted many bad instances in the first iteration with high confidence, like `"shelled gastropods are herbivores"` and `"bivalves are filter feeders"`. Later, invertebrate suffered a fate similar to the "arthropod" category when CMC learned the prefix `"beetle"` and suffix `"beetles"`.
- **mountainRange**: Similar to the "invertebrate" category, CSEAL extracted many high-confidence incorrect instances in the first iteration, such as `"panbiogeography"`, `"list of basic geography topics"`, and `"mike 3"`. Later, CMC learned the suffix `"mountains"`

which led to promotions like `"what plants grow in the mountains"`.

- **movie**: In iteration 55, CMC learned the prefix `"contact"` which led to promotions like `"contact customer support"` and `"contact the director"`.
- **plant**: CMC learned the suffix `"flowers"` which led to `"deliver flowers"`, `"online flowers"`, and `"pictures flowers"`.

B.2 Relations

- **cityCapitalOfState**: Many correct "cityLocatedInState" instances (cities that are located in states but are not capital cities) were promoted for this relation. The key problem is that there is not a good source of near-miss negative examples available from a mutually exclusive relation in the ontology (i.e., one that would provide `("Dallas", "Texas")` as a negative example for "cityCapitalOfState").
- **newspaperInCity**: Incorrect patterns learned by CPL (e.g., `"Y Times and X"` and `"Y Stock Exchange and X"`) led to many incorrect instances where newspapers were thought to be based in `"New York"`.
- **stateLocatedInCountry**: Poor type-checking led to incorrect promotions like `("Air Force", "U.S.")` and `("DOE", "U.S.")`. Adding a "governmentOrganization" category could prevent these first arguments from being allowed as cities.
- **teamPlaysInLeague**: A rule was learned in the 40th iteration that concluded that all teams that played football played in the league `"NFL"`. This led to many incorrect promotions like `"Aggies"` (a college football team) and `"Brazil"` (a country with a national soccer team).
- **productInstanceOf**: Overly general patterns and poor type-checking led to incorrect instances being promoted, like `("Microsoft Office", "PC")` and `("Adobe Illustrator", "computer")`.

Bibliography

Steven Abney. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367, 2002.

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.

J. R. Anderson, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1050, 2004.

Rie Kubota Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

Maria-Florina Balcan and Avrim Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT)*, pages 111–126, 2004.

Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

Michele Banko and Oren Etzioni. Strategies for lifelong knowledge extraction from the web. In *Proceedings of the 4th International Conference on Knowledge Capture*, pages 95–102, 2007.

Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, pages 28–36, 2008.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1):211–231, February 1999.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational learning theory*, pages 92–100, 1998.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

Matthew Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Comput.*, 11(5):1155–1182, 1999.

Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology*, pages 172–183, 1998.

Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300, 1994.

Razvan C. Bunescu and Raymond J. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583, 2007.

Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63(2):129–156, 1994.

Michael J. Cafarella, Alon Y. Halevy, Daisy Z. Wang, Eugene W. 0002, and Yang Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. In *WebDB*, 2008.

Michael J. Cafarella, Jayant Madhavan, and Alon Halevy. Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61, 2008.

J. Callan and M. Hoy. Clueweb09 data set. http://boston.lti.cs.cmu.edu/Data/clueweb09/, 2009.

Jaime Carbonell, Oren Etzioni, Yolanda Gil, Robert Joseph, Craig Knoblock, Steve Minton, and Manuela Veloso. PRODIGY: an integrated architecture for planning and learning. *SIGART Bull.*, 2(4):51–55, 1991.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, 2010.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M.

Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010.

Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. *Machine Learning*, (28):41–75, 1997.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 280–287, 2007.

William W. Cohen, Matthew Hurst, and Lee S. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Proceedings of the 11th International Conference on World Wide Web*, pages 232–241, 2002.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, 2008.

James R. Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007.

Hal Daumé, III. Cross-task knowledge-constrained self training. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 680–688, 2008.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, 1997.

Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1034–1041, 2005.

Doug Downey, Matthew Broadhead, and Oren Etzioni. Locating complex named entities in web text. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2733–2739, 2007.

Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602, 2008.

Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.

L.D. Erman, F. Hayes-Roth, V. Lesser, and D.R. Reddy. The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, 1980.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, June 2005.

Kuzman Ganchev, Joao Graca, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. Technical Report MS-CIS-09-16, University of Pennsylvania Department of Computer and Information Science, 2009.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17*, pages 529–536, 2005.

Zellig S. Harris. Distributional structure. *Word*, 10(2–3):775–793, 1954.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, pages 539–545, 1992.

Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, pages 268–275, 1990.

Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff. Toward completeness in concept extraction and classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 948–957, 2009.

Nicholas Kushmerick. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

J.E. Laird, A. Newell, and P.S. Rosenbloom. SOAR: An architecture for general intelligence. *Artif. Intel.*, 33:1–64, 1987.

Pat Langley, Kathleen B. McKusick, John A. Allen, Wayne F. Iba, and Kevin Thompson. A design for the ICARUS architecture. *SIGART Bull.*, 2(4):104–109, 1991.

Douglas B. Lenat. Eurisko: A program that learns new heuristics and domain concepts. *Artif. Intel.*, 21(1-2):61–98, 1983.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

Qiuhua Liu, Xuejun Liao, and Lawrence Carin. Semi-supervised multitask learning. In *Advances in Neural Information Processing Systems*, 2008.

Jayant Madhavan, David Ko, Lucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy. Google's deep web crawl. *Proc. VLDB Endow.*, 1(2):1241–1252, 2008.

Gideon S. Mann and Andrew McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*, pages 593–600, 2007.

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, 2009.

Tom M. Mitchell, John Allen, Prasad Chalasani, John Cheng, Oren Etzioni, Marc N. Ringuette, and Jeffrey C. Schlimmer. Theo: A framework for self-improving systems. *Arch. for Intelligence*, pages 323–356, 1991.

Tom M. Mitchell. Generative and discriminative classifiers: Naive bayes and logistic regression. `http://www.cs.cmu.edu/~tom/mlbook/NbayesLogReg-2-05.pdf`, 2006. Unpublished book chapter.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Journal of Linguisticae Investigationes*, 30(1), 2007.

David Nadeau, Peter D. Turney, and Stan Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Proceedings of the Canadian Conference on Artificial Intelligence*, 2006.

David Nadeau. *Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision*. PhD thesis, University of Ottawa, November 2007.

Kamal Nigam, Andrew Mccallum, and Tom Mitchell. Semi-supervised text classification using EM. In *Semi-Supervised Learning*, pages 33–56. MIT Press, 2006.

Eric Normand, Kevin Grant, Elias Ioup, and John Sample. Improving relation extraction by exploiting properties of the target relation. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, pages 553–561, 2009.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 809–816, 2006.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, 2002.

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana M. Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947, 2009.

Marco Pennacchiotti and Patrick Pantel. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 238–247, 2009.

J. Ross Quinlan and R. Mike Cameron-Jones. Foil: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20, 1993.

Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479, 1999.

Tony Rose, Mark Stevenson, and Miles Whitehead. The reuters corpus volume 1 - from yesterdays news to tomorrows language resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31, 2002.

Benjamin Rosenfeld and Ronen Feldman. Using corpus statistics on entities to improve semi-supervised relation extraction from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

Sunita Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311, 2006.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the Workshop on Learning with Multiple Views*, 2005.

David A. Smith and Jason Eisner. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 667–677, 2007.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

Charles Sutton and Andrew Mccallum. Introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

Partha Pratim Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. A context pattern induction method for named entity extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 141–148, 2006.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

S. Thrun and T. Mitchell. Lifelong robot learning. In *Robotics and Autonomous Systems*, volume 15, pages 25–46, 1995.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646, 1996.

Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, pages 491–502, 2001.

Nicola Ueffing. Self-training for machine translation. In *NIPS Workshop on Machine Learning for Multilingual Information Access*, 2006.

Benjamin Van Durme and Marius Paşca. Finding cars, goddesses and enzymes: parametrizable acquisition of labeled instances for open-domain information extraction. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 1243–1248, 2008.

Vladimir N. Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 342–350, 2007.

Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, 2008.

Richard C. Wang and William W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1512, 2009.

Casey Whitelaw, Alex Kehlenbeck, Nemanja Petrovic, and Lyle Ungar. Web-scale named entity recognition. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 123–132, 2008.

Roman Yangarber. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 343–350, 2003.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296, March 2009.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003.

Xiaojin Zhu. Semi-supervised learning literature survey. Computer Sciences Technical Report 1530, University of Wisconsin–Madison, 2008.