Carnegie Mellon University
Machine Learning Department

Thesis Proposal

Coupled Semi-Supervised Learning

Andrew Carlson

Thesis Committee:
Tom Mitchell (Chair)
William Cohen
Noah Smith
Oren Etzioni (University of Washington)

May 12, 2009

Abstract

Semi-supervised machine learning methods have shown promise toward allowing the learning of accurate classifiers with much less human effort than traditional supervised methods. However, semi-supervised learning does not always help, and can often even harm the performance of the learned model. This thesis argues that the effectiveness of semi-supervised learning can be greatly 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 be a much more constrained, and therefore easier, set of learning problems.

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. We discuss preliminary work on a system that demonstrates the benefit of coupled semi-supervised learning of 30 category and relation classifiers when run on 200 million web pages. We then present proposed work that explores how to run this system indefinitely, how to better model the world that is discussed on the web, and theoretical analysis of coupled semi-supervised learning.

1 Introduction

Great successes have been achieved with supervised machine learning methods. Supervised methods train a model from a pool of labeled examples of the 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 classifying medical images. However, the application of machine learning methods is greatly constrained by the lack of availability of large labeled training sets. This motivates the development of semi-supervised methods, which can exploit unlabeled data in addition to labeled data.

Some of the earliest successes in semi-supervised learning came from bootstrap learning methods (also referred to as 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 showed promise, but suffered from issues of divergence, 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 divergence, it could yield a significant step forward for semi-supervised learning.

One approach to forestalling divergence 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 divergence still occurs.

This work explores the idea that adding many different constraints beyond the mutual exclusion constraint will enable more accurate semi-supervised learning.

2 Thesis

The central thesis of this work is that it is possible to learn many accurate function approximators with relatively little labeled data by leveraging bootstrap 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.

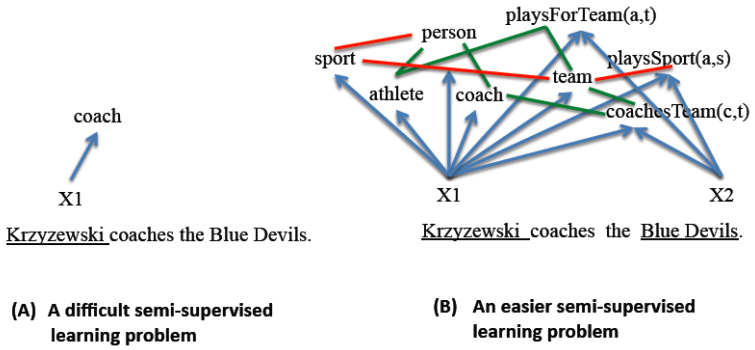

Figure 1: In semi-supervised learning, coupling the training of many classifiers for entities and relations (B) results in a more constrained, and therefore easier learning problem than training a single classifier (A).

This idea is illustrated in Figure 1. Here arrows represent functions being learned, and lines represent constraints between the outputs of functions being learned. At first glance, problem (B) seems like the harder learning problem, because it looks much more complex. This thesis pursues the idea 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, $\chi_i : \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_m \rightarrow \{0, 1\}$. 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 $\mathcal{D}$. 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.

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

- A vast collection 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. IsACity($\cdot$), IsMayorOf($\cdot, \cdot$)) and a rich network of constraints between them (e.g. $City(x) \rightarrow Location(x)$, $IsMayorOf(x, y) \rightarrow City(x) \wedge Person(y), City(x) \rightarrow \neg Country(x)$).
- 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 2-tuple of some smaller space X. Let $f_1(x_1, x_2)$ be the IsAPerson(x_1) predicate (which ignores the value of x_2), $f_2(x_1, x_2)$ the IsCity(x_2) predicate, and $f_3(x_1, x_2)$ the IsMayorOf(x_1, x_2) predicate. We could perform type checking of f_3 by defining two constraint functions. $\chi_1(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2))$ would be 1 if $f_3(x_1, x_2) \rightarrow f_1(x_1, x_2)$ is satisfied, and 0 otherwise. $\chi_2(f_1(x_1, x_2), f_2(x_1, x_2), f_3(x_1, x_2))$ would be 1 if $f_3(x_1, x_2) \rightarrow f_2(x_1, x_2)$ is satisfied, and 0 otherwise.

3 Related Work

3.1 Multitask Learning

Work on multitask learning has 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]. A common approach has been to use a neural network with an output unit for each function being learned, and shared hidden units that can leverage shared structure between the functions. A very large neural network was trained to perform many language processing tasks in a multitask fashion by Collobert and Weston [2008]. Their results demonstrated that joint learning yielded significantly better performance. Supervised training was performed, except for some semi-supervised training of a language model. A semi-supervised approach to multitask learning by Liu et al. [2008] assumed that the tasks to be learned are related, in that they should have similar parameters. Thus a prior was used that encouraged the different tasks to share parameters.

Ando et al. [2005] learns predictors for many different tasks on purely unlabeled data, and then, in a supervised setting, learns how those predictors transfer over to some target problem. This approach allows the learner to learn how the supervised task is related to the unsupervised tasks. In a supervised setting, their framework allows multiple tasks to be structurally related, an idea reminiscent of the previously discussed work on multitask learning.

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 sophisticated relationships than 'these functions should share features' or 'these functions should have similar parameters.' We believe that there are many more ways that tasks can be 'related.' We also aim to push forward the relatively unexplored area of semi-supervised multitask learning.

3.2 Bootstrap Learning

Bootstrapping learning approaches (which are also called self-training) 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. The model is then retrained, 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 web page classification [Blum and Mitchell, 1998], named entity classification [Collins and Singer, 1999], parsing [McClosky *et al.*, 2006], and machine translation [Ueffing, 2006].

Bootstrapping approaches to information extraction can yield impressive results with little initial human effort [Brin, 1998; Agichtein and Gravano, 2000; Pasca *et al.*, 2006]. However, after many iterations, they usually suffer from semantic drift, where errors in labeling during the bootstrapping process accumulate and the learned concept 'drifts' from what was initially intended [Curran *et al.*, 2007]. Coupling the learning of different predicates by using positive examples of one predicate as negative examples for other predicates has been shown in some cases to help limit this drift [Riloff and Jones, 1999; Yangarber, 2003]. Additionally, ensuring that relation arguments are of a certain, expected type can help mitigate the promotion of incorrect instances [Paşca *et al.*, 2006; Rosenfeld and Feldman, 2007].

3.3 Fact Extraction from the Web

Our approach to information extraction is based on using high precision contextual patterns (e.g., 'is mayor of X' suggests that X is a city). An early pattern-based approach to information extraction acquired 'is a' relations from text using generic contextual patterns [Hearst, 1992]. This approach was later scaled up to the web by Etzioni et al. [2005]. Brin [1998] learned patterns to extract instances of relations from the web and demonstrated it with the AuthorOf(author, book) relation.

Results for relation extraction systems have been improved by checking the types of the arguments. One previous approach has been to use existing named entity recognizers or rule-based taggers [Sekine, 2006]. Another is to check that arguments are distributionally similar to seed arguments [Paşca *et al.*, 2006]. The work most similar to ours learns patterns from seed arguments, then heuristically checks candidate arguments by thresholding the number of different patterns an argument occurs with [Rosenfeld and Feldman, 2007].

3.4 Open vs. Closed Information Extraction

Other research explores the task of 'open information extraction,' where the predicates to be learned are not specified in advance [Shinyama and Sekine, 2006; Banko *et al.*, 2007]. Such approaches do not begin with specific relations to extract. Instead, they extract whatever relations they can from a given corpus. In contrast, a key assumption of our work is that the predicates and constraints among them are declared in an ontology which is provided as input to the learner. There are many domains where one could assume that the predicates to be learned are given to the system. For example, in a travel planning domain, where we are populating a knowledge base for use in assisting users with planning trips, it seems reasonable to assume that one could enumerate the relevant predicates (e.g., city, country, airline, airplaneType, price, flightTime, flightNumber).

It seems reasonable to think that the 'open' and 'closed' approaches should be somewhat complementary; if new predicates, constraints, and a few high-confidence examples for each predicate can be discovered automatically using 'open' methods, then our methods could bootstrap from this knowledge to effectively perform 'open IE.'

3.5 Machine Learning Frameworks for Constrained Semi-Supervised Learning

Recently, there has been interest in using prior knowledge to guide machine learning systems. Chang et al. [2007] present a framework for learning that optimizes the sum of data likelihood and constraint-based penalty terms, and demonstrate it with semi-supervised learning of segmentation models. Their results demonstrate that prior knowledge like "fields should be separated by punctuation" can be used to improve self-training. Their work shows that constraints can indeed improve semi-supervised training. This thesis argues that this idea can be applied at a large scale to couple the training of hundreds or thousands of models.

Daume [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. This idea is similar to co-training, where multiple views of an instance space exist such that either view is sufficient to label an example [Blum and Mitchell, 1998]. Co-training requires that the functions agree on unlabeled instances. Both of these works demonstrate through theoretical analysis that coupling semi-supervised training can yield more accurate models. We discuss building on this work in our proposed work.

4 Preliminary Work - Coupling Semi-supervised Learning of Categories and Relations

As preliminary work toward this thesis, we have built a system called Coupled Bootstrap Learner (CBL) [Carlson *et al.*, 2009]. CBL learns to classify noun phrases (NPs) as to whether or not they are members of different categories and ordered pairs of NPs as to whether or not certain relations hold between those pairs of NPs. It does so using semi-supervised learning methods that self-train on newly self-labeled data. The learning of different functions is coupled by enforcing that new data is labeled in a manner consistent with the various known constraints.

Our preliminary work with CBL shows that using relatively simple models for assessment and an iterative bootstrap learning algorithm, coupling the learning of many functions does indeed yield high accuracy results. Additionally, in implementing CBL, we have created a platform upon which this thesis work can build. Extensions are discussed later in this proposal in Section 5.

4.1 Approach

4.1.1 Terminology

A *category* is a single-argument predicate. A *relation* is a two-argument predicate. Relations can be *symmetric*, meaning that the relation is unordered, and that $R(arg1, arg2) \Leftrightarrow R(arg2, arg1)$. Relations can also be *antisymmetric*, meaning that $R(arg1, arg2) \Rightarrow \neg R(arg2, arg1)$. An *instance* of a predicate is a noun phrase or pair of noun phrases. Instances can be *positive* or *negative* with respect to a specific predicate. A *pattern* is a string of tokens with either one or two placeholders for arguments (e.g. 'game against _' and 'arg1 , head coach of arg2') . A *promoted instance* is an instance which our algorithm believes to be a positive instance of some predicate. A *promoted pattern* is a pattern believed to be a high-precision extractor for some predicate.

Algorithm 1: CBL Algorithm

Input: An ontology $\mathcal{O}$, and text corpus C
Output: Trusted instances/patterns for each predicate

SHARE initial instances/patterns among predicates;
for $i = 1, 2, \ldots, \infty$ **do**
  foreach *predicate* $p \in \mathcal{O}$ **do**
    EXTRACT new candidate instances/patterns;
    FILTER candidates;
    TRAIN instance/pattern classifiers;
    ASSESS candidates using trained classifiers;
    PROMOTE highest-confidence candidates;
  end
  SHARE promoted items among predicates;
end

4.1.2 Algorithm Description

The inputs to CBL consist of a large corpus of unlabeled text, and an initial *ontology* $\mathcal{O}$ with pre-defined categories and relations. Six types of information are specified by the ontology for each predicate:

1. A set of seed positive instances of that predicate (e.g. for 'city', seeds might be 'Chicago' and 'Paris')

2. Optionally, a small set of patterns that are known to yield high-precision extraction (e.g. 'cities such as _')

3. A list of other predicates in the ontology with which that predicate is mutually exclusive (e.g. 'city' is mutually exclusive with 'food')

4. A list of other predicates of which that predicate is a subset (e.g. 'city' is a subset of 'location')

5. For category predicates, the ontology specifies whether category instances must be proper nouns, common nouns, or whether they can be either (e.g. instances of 'city' are proper nouns).

6. For relations, the ontology may specify the argument types in terms of categories defined elsewhere in the ontology (e.g., instances of the relation 'headquarteredIn' are pairs of noun phrases of type 'company' and 'city').

The goal of CBL is to perform semi-supervised learning of extractors for each category and relation in the ontology, constrained by this initial knowledge in the ontology, with the aim of growing the list of promoted instances for each predicate as much as possible while maintaining high precision. In pursuance of this goal, CBL also incrementally expands the set of trusted patterns for each predicate.

Algorithm 1 gives a summary of the CBL algorithm. First, seed instances and patterns are shared among predicates using the known mutual exclusion and subset relations. Then, for an

indefinite number of iterations, we expand the sets of promoted instances and patterns for each predicate. The steps of this loop are explained in detail in the sections that follow.

CBL was designed and implemented to allow learning many predicates simultaneously from a large sample of text from the web. In each iteration of the algorithm, the information needed from the text corpus is gathered in two passes through the corpus using the MapReduce framework [Dean and Ghemawat, 2008]. This allows us to complete an iteration of the system in 1 hour while working with a corpus of 200 million web pages.

4.1.3 Sharing

At the start of execution, seed instances and patterns are shared among predicates. If predicate A is mutually exclusive with predicate B, A's positive seed instances/patterns become negative instances/patterns for B, and vice versa. If A is a subset of B, A's seed instances/patterns become seeds for B, as well.

The same sharing process happens with promoted instances and patterns at the end of each iteration. Additionally, the arguments for promoted relation instances are also promoted for their respective categories.

4.1.4 Candidate Extraction

CBL finds new candidate instances by using newly promoted patterns to extract the noun phrases that co-occur with those patterns in the text corpus. To keep the size of this set manageable, we limit the number of new candidate instances for each predicate to 1000 by selecting the ones that occur with the most newly promoted patterns. An analogous procedure is used to extract candidate patterns.

Following are our definitions of instances and patterns for categories and relations. We primarily use regular expressions over part-of-speech (POS) tags.

4.1.5 Category Instances

When a promoted category pattern is matched, we look for a noun phrase in the 'blank' of the pattern. Determiners (e.g. 'a', 'the') are ignored. Noun phrases can either be proper or common. Common noun phrases are sequences of nouns and adjectives that end in a noun. Proper noun phrases are sequences of tokens tagged as proper nouns, possibly connected with prepositions, and possibly with common suffixes and prefixes (e.g. 'Mr .' or ', Inc .').

Proper nouns containing prepositions are segmented using a reimplementation of the Lex algorithm [Downey *et al.*, 2007]. The statistics used in the calculations for Lex use a large n-gram database [Brants and Franz, 2006].

4.1.6 Category Patterns

If a promoted category instance is found, we extract the preceding words as a candidate pattern if they match one the following patterns:

- a sequence of verbs followed by a sequence of adjectives, prepositions, or determiners (e.g. 'being acquired by _')

- a sequence of nouns and adjectives followed by a sequence of adjectives, prepositions, or determiners (e.g. 'former CEO of _')

We extract the words following the instance as a candidate pattern if they match one the following patterns:

- a sequence of verbs followed optionally by a noun phrase (e.g. '_ broke the home run record')
- a sequence of verbs followed by a preposition (e.g. '_ said that')

4.1.7 Relation Instances

If a promoted relation pattern (e.g. 'arg1 is mayor of arg2') is found in a sentence, a candidate relation instance is extracted if both placeholders are matched as valid category instances.

4.1.8 Relation Patterns

If both arguments from a promoted relation instance (i.e. a pair of category instances) are found in a sentence then the intervening sequence of words is extracted as a candidate relation pattern if it contains:

- no more than 5 tokens,
- at least one token that is not a stop word,
- at least one uncapitalized word, and
- at least one word that is not a noun.

Additionally, category instances are only extracted if they match the proper/common noun specification for the category, and relation instances are only extracted if both arguments match the specifications for their respective categories.

Candidate extraction is performed for all predicates in a single pass through the corpus using the MapReduce framework. Candidate extraction requires a list of promoted instances and patterns to be used for extraction. The Map task scans each sentence for promoted items using a Trie data structure [Fredkin, 1960]. If one is found, a matching candidate is looked for.

4.1.9 Filtering Candidates

Candidate instances and patterns are filtered to maintain high precision, and to avoid extremely specific patterns. An instance is only considered for assessment if it co-occurs with at least two promoted patterns in the text corpus, and if its co-occurrence count with all promoted patterns is at least three times greater than its co-occurrence count with negative patterns. Candidate patterns are filtered in the same manner using promoted and negative instances.

The co-occurrence counts needed by the filtering step are obtained with an additional pass through the corpus using MapReduce. The Map task is given a list of instances and a list of patterns, and returns the co-occurrence counts for all observed pairs from those lists. This implementation is much more efficient than one that relies on web search queries. CBL typically requires co-occurrence counts of at least 10,000 instances with any of at least 10,000 patterns, which would require 100 million hit count queries.

4.1.10 Assessment

Next, for each predicate in the ontology CBL trains a discretized Naïve Bayes classifier to classify the candidate instances. Its features include pointwise mutual information (PMI) scores [Turney, 2001] of the candidate instance with each of the trusted patterns associated with the class. The current sets of promoted and negative instances are used to train the classifier. Attributes are discretized by splitting the continuous values based on information gain [Fayyad and Irani, 1993].

Patterns are assessed 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 currently being considered, $count(i, p)$ is the co-occurrence count of instance i with pattern p, and $count(p)$ is the hit count of the pattern p. This is a pessimistic estimate because it assumes that the rest of the occurrences of pattern p are not with positive examples of the predicate. We also threshold the denominator by using the 25th percentile candidate pattern hit count for any hit count lower than it. This penalizes extremely rare patterns. This ranking method was inspired by McDowell and Cafarella [2006].

All of the co-occurrence counts needed for the above assessments (for all predicates) are collected at the same time and in the same manner as those required for filtering candidates, as described in the previous section.

4.1.11 Promotion

CBL then ranks the candidates according to their assessment scores and promotes at most 100 instances and 5 patterns for each predicate.

4.2 Experimental Evaluation

We designed our experimental evaluation to try to answer the following questions:

- Can CBL iterate many times and still achieve high precision?
- How helpful are the types of coupling that we employ?
- Can we extend existing semantic resources?

4.2.1 Configurations of the Algorithm

We ran our algorithm in three different configurations:

- Full: The algorithm as described in Section 4.1.2.
- No Sharing Among Same-Arity Predicates (NS): The full algorithm, except that predicates of the same arity do not share promoted instances and patterns with each other. Seed instances and patterns *are* shared, though, so each predicate has a small, fixed pool of negative evidence. Note that training of categories remains coupled to training of relations through constraints on the argument types of the relations.

Predicate	5 iterations			10 iterations			15 iterations		
	Full	NS	NCR	Full	NS	NCR	Full	NS	NCR
Actor	93	100	100	93	97	100	100	97	100
Athlete	100	100	100	100	93	100	100	73	100
Board Game	93	76	93	89	27	93	89	30	93
City	100	100	100	100	97	100	100	100	100
Coach	100	63	73	97	53	43	97	47	47
Company	100	100	100	97	90	97	100	90	100
Country	60	40	60	30	43	27	40	23	40
Economic Sector	77	63	73	57	67	67	50	63	40
Hobby	67	63	67	40	40	57	20	23	30
Person	97	97	90	97	93	97	93	97	93
Politician	93	93	97	73	53	90	90	53	87
Product	97	87	90	90	87	100	97	90	77
Product Type	93	93	90	70	73	97	77	80	67
Scientist	100	90	97	97	63	97	93	60	100
Sport	100	90	100	93	67	83	97	27	90
Sports Team	100	97	100	97	70	100	90	50	100
Category Average	92	84	89	82	70	84	83	63	79
Acquired(Company, Company)	77	77	80	67	80	47	70	63	47
CeoOf(Person, Company)	97	87	100	90	87	97	90	80	83
CoachesTeam(Coach, Sports Team)	100	100	100	100	100	97	100	100	90
CompetesIn(Company, Econ. Sector)	97	97	80	100	93	67	97	63	60
CompetesWith(Company, Company)	93	80	60	77	70	37	70	60	43
HasOfficesIn(Company, City)	97	93	40	93	90	27	93	57	30
HasOperationsIn(Company, Country)	100	95	50	100	97	40	90	83	13
HeadquarteredIn(Company, City)	77	90	20	70	77	27	70	60	7
LocatedIn(City, Country)	90	67	57	63	50	43	73	50	30
PlaysFor(Athlete, Sports Team)	100	100	0	100	97	7	100	43	0
PlaysSport(Athlete, Sport)	100	100	27	93	80	10	100	40	30
TeamPlaysSport(Sports Team, Sport)	100	100	77	100	97	80	93	83	67
Produces(Company, Product)	91	83	90	83	93	67	93	80	57
HasType(Product, Product Type)	73	63	17	33	67	33	40	57	27
Relation Average	92	88	57	84	84	48	84	66	42
All	92	86	74	83	76	68	84	64	62

Table 1: Precision (%) for each predicate. Results are presented after 5, 10, and 15 iterations, for the Full, No Sharing (NS), and No Category/Relation Coupling (NCR) configurations of CBL .

- No Category/Relation coupling (NCR): The full algorithm, except that relation instance arguments are not filtered or assessed using their specified categories, and arguments of promoted relations are not shared as promoted instances of categories. This decouples the training of relations from training of categories, except that relations use the common/proper noun specifications of their arguments to filter out implausible instances. Sharing among predicates with the same arity still occurs.

4.2.2 Initial ontology

Our ontology contained categories and relations related to two domains: companies and sports. Extra categories were added to provide negative evidence to the domain-related categories: 'hobby' for 'economic sector', 'actor,' 'politician,' and 'scientist' for 'athlete' and 'coach', and 'board game' for 'sport'. Table 1 lists each predicate in the leftmost column. Categories were started with 10–20 seeds and 5 seed patterns. The seeds were specified by a human, and the seed patterns were derived from the generic patterns of Hearst for each predicate [Hearst, 1992]. Relations were started with similar numbers of seed instances, and no seed patterns (it is less obvious how to generate good seed patterns from relation names).

4.2.3 Corpus

Our text corpus was from a 200-million page web crawl. We parsed the HTML, filtered out non-English pages by using a stop word ratio threshold, then filtered out web spam and adult content using a 'bad word' list. The pages were then segmented into sentences, tokenized, and tagged with parts-of-speech using the OpenNLP package. Finally, we filtered the sentences to eliminate sentences 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).

4.2.4 Experimental Procedure

We ran each configuration for 15 iterations. To evaluate the system's performance, we sampled 30 instances from the total promoted set for each predicate in each configuration after 5, 10, and 15 iterations, pooled together all of the samples for each predicate, and then judged their correctness. The judge did not know which run an instance was sampled from. We estimated the precision of the promoted instances from each run after 5, 10, and 15 iterations as the number of correct promoted instances divided by the number sampled.

4.2.5 Experimental Results

Table 1 shows the precision of each of the three algorithm configurations for each category and relation predicate, after 5, 10, and 15 iterations. As is apparent in this table, fully coupled training (Full) outperforms training when coupling is removed between categories and relations (NCR), and also when coupling is removed among predicates of the same arity (NS). The net effect is substantial, as is apparent from the bottom row of Table 1, which shows that the precision of Full outperforms NS by 6% and NCR by 18% after the first 5 iterations, and by an even larger 20% and 22% after 15 iterations. This increasing gap in precision as iterations increase reflects the ability of coupled learning to constrain the system to reduce the otherwise common drift associated with self-trained classifiers.

Configuration	Categories Instances	Prec.	Relations Instances	Prec.
Full	970	83	191	84
NS	1337	63	307	66
NCR	916	79	458	42

Table 2: Average numbers of promoted category and relation instances and estimates of their precision for each configuration of CBL after 15 iterations.

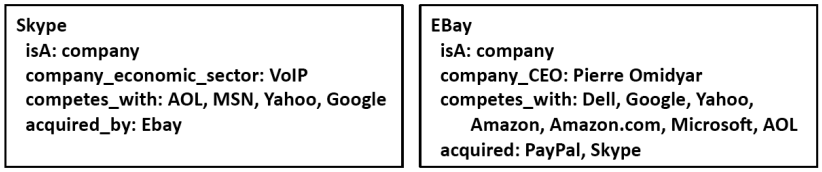

Figure 2: Extracted facts for two companies discovered by CBL Full. These two companies were extracted by the learned 'Company' extractor, and the relations shown were extracted by learned relation extractors.

Using Student's paired t-test, we found that for categories, the difference in performance between Full and NS is statistically significant after 5, 10, and 15 iterations (p-value < 0.05).[1] No significant difference was found between Full and NCR for categories, but this is not a surprise, because NCR still uses mutually exclusive and subset constraints. The same test finds that the differences between Full and NS are significant for relations after 15 iterations, and the differences between Full and NCR are significant after 5, 10, and 15 iterations for relations.

The worst-performing categories after 15 iterations of Full are 'country,' 'economic sector,' and 'hobby.' The Full configuration of CBL promoted 1637 instances for 'country,' far more than the number of correct answers. Many of these are general geographic regions like 'Bayfield Peninsula' and 'Baltic Republics.' In the 'hobby' case, promoting patterns like 'the types of _' led to the category drifting into a general list of plural common nouns. 'Economic sector' drifted into academic fields like 'Behavioral Science' and 'Political Sciences.' The results of this run can be examined on the web at `http://rtw.ml.cmu.edu/readtheweb.html`. We expect that the learning of these categories would be significantly better if there were even more categories being learned to provide additional negative evidence during the filtering and assessment steps of the algorithm.

At this stage of development, obtaining high recall is not a priority because our intent is to create a continuously running and continuously improving system; it is our hope that high recall will come with time. However, to very roughly convey the completeness of the current results we show in Table 2 the average number of instances promoted for categories and relations for each of the three configurations of CBL after 15 iterations. For categories, not sharing examples results in fewer negative examples during the filtering and assessment steps. This yields more promoted instances on average. For relations, not using type checking yields higher relative recall, but at a much lower level of precision.

Figure 2 gives one view of the type of information extracted by the collection of learned category

[1]Our selection of the paired t-test was motivated by the work of Smucker et al. [2007], but the Wilcoxon signed rank test gives the same results.

Category	Freebase Matches	CBL Instances	Est. Prec.	Est. New Instances
Actor	465	522	100	57
Athlete	54	117	100	63
Board Game	6	18	89	10
City	1665	1799	100	134
Company	995	1937	100	942
Econ. Sector	137	1541	50	634
Politician	74	962	90	792
Product	0	1259	97	1221
Sports Team	139	414	90	234
Sport	134	613	97	461

Table 3: Estimated numbers of 'New Instances', which are correct instances promoted by CBL in the Full 15 iteration run which do not have a match in Freebase, and the values used in calculating them (number of Freebase/CBL matches, number of CBL instances, and the estimated precision of CBL for the predicate).

and relation classifiers. Note the initial seed examples provided to CBL did not include information about either company or any of these relation instances.

4.2.6 Comparison to an Existing Database

To evaluate the capacity of our algorithm to contribute additional facts to publicly available semantic resources, we downloaded complete lists of entries in the Freebase database and estimated how many correct instances of categories we had found that are not in Freebase. Specifically, for the predicates with Freebase entries, we took the promoted items from the Full 15 iteration run, and computed:

$$Precision * |CBLInstances| - |Matches|$$

where $Precision$ is the estimated precision from our random sample of 30 instances, $|CBLInstances|$ is the total number of instances promoted for that predicate, and $Matches$ is the number of promoted instances that had an exact match in Freebase. Table 3 gives these estimates. Many categories with very high precision (e.g. 'actor', 'athlete', 'city', and 'company') had many instances that did not have matches in Freebase. While these numbers are approximate, they indicate a significant potential to use CBL to extend existing semantic resources like Freebase.

5 Proposed Work

5.1 Scaling Up

Our preliminary work has made novel contributions to semi-supervised learning and information extraction by showing that coupling the learning of many functions yields more accurate models. Two ways to extend this work are to:

- Add more coupling constraints.
- Learn more functions.

5.1.1 Adding Coupling Constraints

Adding more coupling constraints should further constrain our learning problems and yield even more accurate training. One idea for new constraints is to induce probabilistic rules from current beliefs using a rule learner similar to FOIL [Quinlan and Cameron-Jones, 1993].

For example, we can induce Horn clause rules from the output of CBL such as these:

0.84 playsSport(?x,?y) :- playsFor(?x,?z), teamPlaysSport(?z,?y)
0.70 playsSport(?x,baseball) :- playsFor(?x,yankees)

We foresee that such learned rules will play two important roles in our future work. First, these rules can be used to infer new relation instances that were not extracted directly from the text, but which follow from other extracted instances. Second, each rule defines a new probabilistic coupling constraint among the relations it refers to. If MBL were to invoke the rule learner during each iteration, it could use these rules both as an additional source of proposed predicate instances, and to grow iteratively the set of coupling constraints it uses to achieve high accuracy.

5.1.2 Adding More Functions

As discussed in Section 4.2.5, many of the errors made by CBL are the result of semantic drift. For example, the 'country' classifier promotes many geographical regions that are not countries or cities. The 'economic sector' classifier promotes many academic fields such as 'Behavioral Science.'

If we were learning hundreds of predicates rather than tens, each individual predicate would be much more constrained during training. Thus, dramatically extending our ontology would be a promising direction for future work.

5.2 Probabilistic Models

We see two key opportunities in the direction of devising better probabilistic models to improve our current work. The first idea is to use unlabeled data to better estimate the parameters of our current models in the CBL algorithm. This would be an incremental change, but could allow significantly better assessment in CBL. The second idea is to create a global probabilistic model at the entity level, which can represent notions of synonymy and ambiguity. Searching for model parameters which best explain the observed evidence would provide an approach justified by an objective function (it is unclear exactly what objective our bootstrapping approach is currently optimizing).

5.2.1 Exploiting Unlabeled Data for Assessment in CBL

In the current implementation of CBL, promoted instances and patterns are used to probabilistically assess candidate instances and patterns. The instance classifier for a predicate uses promoted patterns as features and promoted instances as examples. We could extend this approach by:

- Using additional patterns as features (ones that have not been promoted in the self-supervised bootstrapping process)
- Using additional instances to train the assessor with EM. This would let us perform semi-supervised training using unlabeled instances.

When assessing candidate patterns, we use the promoted instances it occurs with to estimate the precision of a candidate pattern. We currently use the sum of co-occurrence counts with promoted instances divided by the hit count of the pattern. This is a pessimistic estimate of the true precision of the pattern, because it assumes that all of the occurrences that were not with promoted instances are with incorrect instances. Instead, we could label the unpromoted instances that each candidate pattern occurs with, using the instance assessor, and use those assessment scores to estimate the precision of the pattern.

These pieces of proposed work also serve to highlight the fact that the models used by CBL in the preliminary work were baseline placeholders, but still yielded promising results.

5.2.2 An Entity-Level Global Probabilistic Model

CBL currently learns to classify noun phrases or pairs of noun phrases. 'Carnegie Mellon' and 'CMU' are viewed as entirely separate concepts. This is unrealistic both in our case study as well as in general— one would not model objects in the world as unique assignments to a grid of pixels in a computer vision application, either. In order to more accurately model the world, we should work at the entity level, rather than the token level. 'Carnegie Mellon' and 'CMU' should be different ways of naming the same entity, *CarnegieMellonUniversity*. Entities could be modeled as clusters of noun phrases, each of which could potentially refer to that entity.

To cluster noun phrases, Kok and Domingos [2008] used an unsupervised approach that optimized the likelihood of observed (noun phrase, relation, noun phrase) triples. A prior over the number of clusters and number of elements in a cluster forced optimal solutions to the problem of optimizing the object to find clusters that explained the observed data. Yates and Etzioni [2007] used a different unsupervised approach to modeling the probability that two strings co-refer based on their textual similarity and the similarity of the contexts which which they each co-occur.

These methods give good results for large-scale synonym resolution. However, they have two shortcomings. First, they are not designed to allow the use of labeled data when it is available. We have seed knowledge which should be very useful to our model estimation task. Secondly, these methods do not explicitly model ambiguity. For example, 'CMU' should be a potential name for the entity *CarnegieMellonUniversity*, but also for *CentralMichiganUniversity*. In these clustering approaches, 'CMU' would end up in a single cluster.

We suspect that better performance can come by modeling the world as a collection of entities for which some relations hold. Noun phrases would be text strings that can refer to entities, but the same noun phrase could refer to multiple entities. A prior that encourages most proper noun phrases to refer to just a few unique entities might yield a reasonable model.

This direction of work has great potential, as progress towards modeling 'the world' as a collection of entities with relationships among them would be relevant to many problems other than fact extraction from the web. Doing so in a scalable way that could cope with the size of the web would be a significant contribution.

5.3 Theoretical Analysis of Coupled Semi-Supervised Learning

We would like to develop a theoretical framework to allow us to analyze coupled semi-supervised learning. Questions we would like to explore include: How does the sample complexity of learning accurate function approximations relate to the number of functions being learned? How can we

characterize how two functions are coupled? And, what is the relationship between the accuracy of a set of classifiers and them being consistent with some constraints?

One promising direction would build on the work of Daume [2008]. His work considers cross-task learning of 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:

$$Pr[\chi(f_1(x), h_1(x))) = 1]^{-1}$$

It would be interesting to try to build on this framework and the analysis for the case where we have many more functions and many more constraints.

Another direction of research is to analyze our coupled semi-supervised learning approaches in the PAC-style framework of Balcan and Blum [2005]. If one considers the concept class $\mathcal{C}$ to be made up of many different subconcepts (each representing one of our functions to be learned), then a compatibility function χ would give a measure of how well the coupling constraints are satisfied on unlabeled data drawn from a data distribution D.

6 Schedule

- May 2009: Proposal
- Background Process: Theoretical analysis
- Summer 2009: Integrate rule learner and scale up to more predicates
- Fall 2009: Model design, experimental work from rule learner
- Spring 2010: Experimental work with probabilistic models
- Summer 2010: Thesis writing
- September 2010: Graduation

References

[Agichtein and Gravano, 2000] 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.

[Ando *et al.*, 2005] 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.

[Banko *et al.*, 2007] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.

[Brants and Franz, 2006] Thorsten Brants and Alex Franz. Web 1t 5-gram version 1, 2006.

[Brin, 1998] 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.

[Carlson *et al.*, 2009] Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, 2009.

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

[Chang *et al.*, 2007] Ming-Wei Chang, Lev-Arie Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, 2007.

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

[Collobert and Weston, 2008] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[Curran *et al.*, 2007] James R. Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *PACLING*, 2007.

[Daumé, 2008] Hal Daumé. Cross-task knowledge-constrained self training. In *EMNLP*, pages 680–688, Honolulu, Hawaii, 2008.

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

[Downey *et al.*, 2007] Doug Downey, Matthew Broadhead, and Oren Etzioni. Locating complex named entities in web text. In *IJCAI*, 2007.

[Etzioni *et al.*, 2005] 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. *Artif. Intell.*, 165(1):91–134, June 2005.

[Fayyad and Irani, 1993] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *UAI*, pages 1022–1027, 1993.

[Florina Balcan and Blum, 2005] Maria Florina Balcan and Avrim Blum. A PAC-style model for learning from labeled and unlabeled data. In *COLT*, pages 111–126, 2005.

[Fredkin, 1960] Edward Fredkin. Trie memory. *Commun. ACM*, 3(9):490–499, September 1960.

[Hearst, 1992] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545, 1992.

[Kok and Domingos, 2008] Stanley Kok and Pedro Domingos. Extracting semantic networks from text via relational clustering. In *ECML/PKDD*, pages 624–639, 2008.

[Liu *et al.*, 2008] Qiuhua Liu, Xuejun Liao, and Lawrence Carin. Semi-supervised multitask learning. In *NIPS*, 2008.

[McClosky *et al.*, 2006] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *NAACL*, pages 152–159, 2006.

[McDowell and Cafarella, 2006] Luke K. McDowell and Michael Cafarella. Ontology-driven information extraction with ontosyphon. In *ISWC*. Springer, 2006.

[Paşca *et al.*, 2006] Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: fact extraction in the fast lane. In *ACL*, pages 809–816, 2006.

[Pasca *et al.*, 2006] Marius Pasca, 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 *AAAI*, 2006.

[Quinlan and Cameron-Jones, 1993] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *ECML*, 1993.

[Riloff and Jones, 1999] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*, pages 474–479, 1999.

[Rosenfeld and Feldman, 2007] Benjamin Rosenfeld and Ronen Feldman. Using corpus statistics on entities to improve semi-supervised relation extraction from the web. In *ACL*, pages 600–607, 2007.

[Sekine, 2006] Satoshi Sekine. On-demand information extraction. In *ACL*, pages 731–738, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[Shinyama and Sekine, 2006] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL06*, pages 304–311, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[Smucker *et al.*, 2007] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, 2007.

[Thrun, 1996] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, pages 640–646, 1996.

[Turney, 2001] P. D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *EMCL*, pages 491–502, London, UK, 2001.

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

[Yangarber, 2003] Roman Yangarber. Counter-training in discovery of semantic patterns. In *ACL*, pages 343–350, 2003.

[Yarowsky, 1995] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196, 1995.

[Yates and Etzioni, 2007] Alexander Yates and Oren Etzioni. Unsupervised resolution of objects and relations on the web. In *HLT-NAACL*, 2007.