Coupled Semi-Supervised Learning for Information Extraction

Andrew Carlson
Schoool of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
acarlson@cs.cmu.edu

Justin Betteridge
Schoool of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jbetter@cs.cmu.edu

Richard C. Wang
Schoool of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
rcwang@cs.cmu.edu

Estevam R. Hruschka Jr.
Federal University of Sao Carlos
Sao Carlos, SP - Brazil
estevam@dc.ufscar.br

Tom M. Mitchell
Schoool of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
tom.mitchell@cs.cmu.edu

ABSTRACT

We consider the problem of semi-supervised learning to extract categories (e.g., academic fields, athletes) and relations (e.g., PlaysSport(athlete, sport)) from web pages, starting with a handful of labeled training examples of each category or relation, plus hundreds of millions of unlabeled web documents. Semi-supervised training using only a few labeled examples is typically unreliable because the learning task is underconstrained. This paper pursues the thesis that much greater accuracy can be achieved by further constraining the learning task, by coupling the semi-supervised training of many extractors for different categories and relations. We characterize several ways in which the training of category and relation extractors can be coupled, and present experimental results demonstrating significantly improved accuracy as a result.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*knowledge acquisition*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis*

General Terms

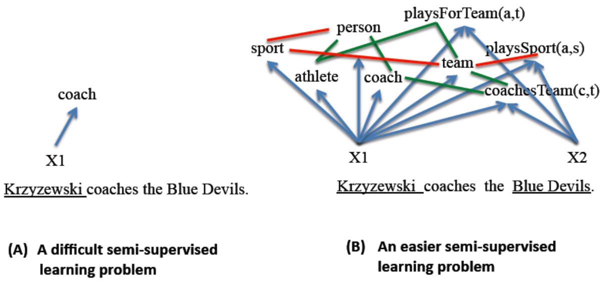
Algorithms, Experimentation

Keywords

Semi-supervised learning, bootstrap learning, information extraction, web mining

1. INTRODUCTION

Machine learning approaches have been shown to be very useful for information extraction from text, including approaches that learn to extract various categories of entities (e.g., Athlete, City) and relations (e.g., CompanyProducesProduct) from structured and unstructured text [3, 28]. 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. Because of this, many researchers have explored *semi-supervised* learning methods that use only a small number of labeled examples of the predicate to be extracted, along with a large volume of unlabeled text [5, 19, 1]. 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.



(A) A difficult semi-supervised learning problem

(B) An easier semi-supervised learning problem

Figure 1: We show that significant improvements in accuracy result from coupling the training of information extractors for many interrelated categories and relations (B), compared with the simpler but much more difficult task of learning a single information extractor (A).

The thesis explored in this paper is that we can achieve much higher accuracy in semi-supervised learning by coupling the simultaneous training of *many* extractors, as suggested in Figure 1. The intuition here is that the underconstrained semi-supervised learning task can be made easier by adding new constraints that arise from coupling the training of many extractors.

WSDM'10, February 4–6, 2010, New York City, New York, USA.

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. We show that our general approach improves accuracies when training both contextual-pattern extractors that extract information from freeform text (e.g., the pattern "mayor of *arg1*" as an extractor for the category City) and wrappers which extract information from semi-structured documents (e.g., the wrapper "`<td class="cty">`*arg1*`</td>`" from some specific URL).

Based on results reported here, we hypothesize that even greater accuracy improvements will be possible by forming a more dense network of inter-constrained learning tasks. Toward this end, we explore two more specific points. First, we identify three general types of coupling among target functions that can be combined to form a dense network of coupled learning problems. Second, we explore the impact of coupling the training of extractors that use freeform text with extractors that leverage semi-structured web pages, based on the intuition that these different techniques should make independent errors.

We believe that the novel contributions of this work are as follows: Our work is the first to couple the simultaneous semi-supervised training of category and relation extractors. It is also the first to couple the training of multiple wrapper inducers by using mutual exclusion and type checking relationships. Finally, this work is the first to couple the *training* (rather than the final outputs) of freeform-text extractors and semi-structured web page wrapper inducers by assuming that they make independent errors, a method that we show provides higher accuracies than using either method alone. More generally, this paper advocates large-scale coupled training as a strategy to significantly improve accuracy in semi-supervised learning, identifies three distinct types of coupling, and experimentally evaluates their utility.

2. RELATED WORK

In this paper, we focus on a "bootstrapping" method for semi-supervised learning. Bootstrapping approaches start with a small number of labeled "seed" examples and iteratively grow the set of labeled examples using high-confidence labels from the current model. Such approaches have shown promise in applications such as web page classification [4], named entity classification [9], parsing [16], and machine translation [24]. Bootstrapping approaches to information extraction can yield impressive results [5, 9, 1]. However, after many iterations, accuracy typically declines because errors in labeling accumulate, a problem that has been called "semantic drift" [10].

To reduce errors introduced in underconstrained semi-supervised learning, several methods have been considered. 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 [19, 27]. Co-Training methods exploit conditionally independent partitions of the feature space to avoid labeling errors [4]. Other non-bootstrapping techniques have used the intuition that different extraction methods should make independent errors to motivate combining predictions from multiple extractors to improve extraction accuracy [22, 6, 18]. Type checking relation arguments using available entity recognizers can help avoid incorrect labels [17, 20]. Our work builds on these different ideas and uses them to couple the simultaneous bootstrapped training of many category and relation extractors in many different ways.

Several machine learning frameworks have been proposed that penalize violations of constraints on unlabeled data [2, 8, 11]. While similar in spirit, our work differs in that we consider using many different kinds of constraints to learn many different functions simultaneously.

In multitask learning, supervised training of "related" functions together can yield higher accuracy than learning them separately [23, 7]. Semi-supervised multitask learning has used a prior that encourages related models to have similar parameters [15]. These methods require that related tasks share similar representations; our work exploits additional ways in which functions can be related, and makes no assumptions regarding similarity of representations across the different functions being learned.

3. COUPLED TRAINING

Central to this work is the idea of coupling the semi-supervised learning of multiple functions to constrain our learning problem. Our method iteratively trains classifiers in a self-supervised manner. It starts by training classifiers using a small amount of labeled data, then uses these classifiers to label unlabeled data. The most confident new labels are added to the pool of data used to train the models (we say that these new labeled examples are *promoted*), and the process repeats. The iterative training is coupled by constraints that restrict allowable candidates and promotions.

3.1 Types of Coupling

We have identified 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 web page, X_1 could represent the words in a page, and X_2 could represent words in hyperlinks pointing to that page (this example was used for the Co-Training setting [4]).

3.2 Coupling Constraints Used in this Paper

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 paper). The general types of coupling discussed above are used to learn these functions in these 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.
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.
3. Unstructured and Semi-structured Text Features: Noun phrases on the web appear in two types of contexts: freeform textual contexts and semi-structured contexts. For example, "Pittsburgh" occurs on the web with a distribution of freeform textual contexts such as "mayor of *arg1*", 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, and that the two distributions are conditionally independent given the class label of the noun phrase. We therefore train two noun phrase classifiers, one using each type of context distribution, and require that the two classifiers agree on the label for each given noun phrase. This is an example of a multi-view constraint.

4. ALGORITHMS

In this section, we present 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. We focus our consideration of predicates on unary relations (categories) and binary relations (ones with two arguments, referred to as *relations* in this paper).

The specific inputs to our algorithms are: a large text corpus and an initial ontology with predefined categories, relations, mutual-exclusion relationships between same-arity predicates, and seed instances for all predicates. Each relation has an ordered pair of argument types, which specify categories that relation instance arguments must be members of. Some additional information is only used by freeform-text extraction methods: seed extraction patterns for categories, and a flag for each category indicating whether instances must be proper nouns, common nouns, or can be either (e.g., instances of City are proper nouns).

The first algorithm, Coupled Pattern Learner (CPL), is a bootstrapping algorithm that leverages mutual-exclusion and type-checking constraints to learn high-precision contextual patterns that are accurate extractors of predicate instances. The second, Coupled SEAL (CSEAL), is a set expansion algorithm that learns "wrappers" to extract instances from semi-structured documents and exploits the same two types of coupling constraints. These two algorithms serve to demonstrate that output and compositional coupling techniques can improve the accuracy of bootstrapped training of multiple types of extractors. The final algorithm, Meta-Bootstrap Learner (MBL), couples the training of subordinate extraction algorithms like CPL and CSEAL using multi-view-agreement constraints.

Algorithm 1: Coupled Pattern Learner (CPL)

```
Input: An ontology $\mathcal{O}$, and text corpus $C$
Output: Trusted 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/patterns;
    PROMOTE top candidates;
  end
end
```

4.1 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., "*arg1* and other software firms" and "*arg1* scored a goal for *arg2*") and uses them to grow a set of high-precision predicate instances. Noun phrases that fill in the "*arg1*" and "*arg2*" 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:

4.1.1 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. 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. It uses part-of-speech tags to segment noun phrases and ignores determiners. Proper noun phrases containing prepositions or conjunctions are segmented using a reimplementation of the Lex algorithm [13]. Category instances are required to obey the proper/common noun specification of the category.

- 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 *arg1*" or "companies acquired by *arg1*") or nouns and adjectives followed by a sequence of adjectives, prepositions, or determiners (e.g., "former CEO of *arg1*"). CPL extracts the words following the instance as a candidate pattern if they are verbs followed optionally by a noun phrase (e.g., "*arg1* broke the home run record"), or verbs followed by a preposition (e.g., "*arg1* said that").
- Relation Instances: If a promoted relation pattern (e.g., "*arg1* is mayor of *arg2*") 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, has a content word, and has an uncapitalized word.

4.1.2 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 three times more than the number of times it co-occurs with patterns from mutually exclusive predicates. This soft constraint is much more tolerant of the inevitable noise in web text as well as ambiguous noun phrases than a hard constraint. Candidate patterns are filtered in the same manner using promoted instances.

4.1.3 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.

4.1.4 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).

Algorithm 2: 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;
      Learn wrappers for each document returned;
      Extract new candidates using wrappers;
    end
    Filter wrappers that extract candidates that
    violate coupling;
    Rank candidate instances;
    Promote top candidates;
  end
end
```

4.1.5 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 [12]. In each iteration of CPL, CPL gathers corpus statistics from this data set 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 5.1.2 for details on the corpus).

4.1.6 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.

4.2 Coupled SEAL

CPL is an example of a semi-supervised text pattern learning algorithm that is aided by coupling. To demonstrate how coupling can improve a different, already implemented extraction algorithm we consider SEAL [26], a wrapper induc-

URL:	`http://www.shopcarparts.com/`
Wrapper:	`.html" CLASS="shopcp">arg1 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/arg1/`
Content:	acura, audi, bmw, buick, cadillac, chevrolet, chrysler, dodge, ford, gmc, honda, hyundai, infiniti, isuzu, ...
URL:	`http://www.hertrichs.com/`
Wrapper:	`<li class="franchise arg1"> <h4><a href="#">`
Content:	buick, chevrolet, chrysler, dodge, ford, gmc, isuzu, jeep, lincoln, mazda, mercury, nissan, pontiac, scion, ...

Table 1: Examples of wrappers constructed by CSEAL for various web pages given the seeds: Ford, Nissan, Toyota. In the table, *arg1* is a placeholder for extracting instances.

Algorithm 3: 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 candidates that were extracted by all
    extractors;
  end
end
```

tion algorithm, and how we can add coupling constraints on top of an existing implementation that we treat as a "black box." First, we will describe the existing algorithm, and then we will describe how we add coupling constraints.

SEAL 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. 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. These context strings are selected to be maximally-long contexts that bracket at least one occurrence of every seed on a page. Table 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 [25]. 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.

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 2. 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 instances for the respective category 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).

After filtering, CSEAL ranks all candidate instances by the number of unfiltered wrappers that extracted them, and promotes the top 100 instances that were extracted by at least two wrappers. To deal with 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 Meta-Bootstrap Learner

Meta-Bootstrap Learner (MBL) couples the training of multiple extraction techniques using a multi-view constraint that requires them to agree. MBL is summarized in Algorithm 3. MBL is based on the intuition that the errors made by different extraction techniques should be independent.

In this paper, 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.

5. EXPERIMENTAL EVALUATION

We designed experiments to explore three main questions: First, does coupling learning using mutual-exclusion and type-checking constraints improve the performance of CPL relative to uncoupled, independent learning using UPL? Second, do mutual-exclusion and type-checking constraints improve the performance of CSEAL relative to the uncoupled methods of SEAL? Finally, does MBL achieve better performance than CPL and CSEAL by combining their outputs with a multi-view constraint?

To answer these questions, we ran CPL, UPL, CSEAL, SEAL, and MBL with CPL 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.

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 paper. We believe that the uncoupled baselines are reasonable and competitive large-scale uncoupled approaches.

5.1 Experimental Methodology

5.1.1 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 2 lists all of the categories in the leftmost column, and Table 3 lists the relations in the leftmost column. Categories were initialized with 15 seed instances and 5 seed patterns. The seed instances were specified by a human, and the seed patterns for each category were derived from the generic patterns of Hearst [14]. 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). Most predicates were declared as mutually exclusive with one another (examples of exceptions include SportsTeam and University; KitchenItem and ProductType; and Company and Product).

5.1.2 Corpus for CPL

The text corpus used by CPL was from a 200-million-page web crawl. 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 "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 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 4.1.5, 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 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.

5.1.3 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 examples provided 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.

5.1.4 General Experimental Procedure

When comparing two algorithms, we ran each algorithm for 10 iterations of bootstrapping, and then assessed the instances promoted by the algorithms. 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 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. We refer to this method of comparison as the *minimum recall* method in the results below. We sampled 30 instances from each of these two sets of instances, and also submitted them to Mechanical Turk.

While samples of 30 instances do not produce tight confidence intervals for individual estimates of precision for a single predicate, they are sufficient for testing for the effects in which we are interested.

CPL can reliably extract the proper case of an instance, 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.

5.1.5 Mechanical Turk Labeling

The various estimates of precision required for our evaluation 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 [21]. 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.

	Precision (%)					Promoted Instances (#)				
Predicate	CPL	UPL	CSEAL	SEAL	MBL	CPL	UPL	CSEAL	SEAL	MBL
AcademicField	70	83	90	97	100	46	903	203	1000	181
Actor	100	33	100	97	100	199	1000	1000	1000	380
Animal	80	50	90	70	97	741	1000	144	974	307
Athlete	87	17	100	87	100	132	930	276	1000	555
AwardTrophyTournament	57	7	53	7	77	86	902	146	1000	79
BoardGame	80	13	70	77	90	10	907	126	1000	31
BodyPart	77	17	97	63	93	176	922	80	1000	61
Building	33	50	30	0	93	597	1000	57	1000	14
Celebrity	100	90	100	100	97	347	1000	72	747	514
CEO	33	30	100	77	100	3	902	322	1000	30
City	97	100	97	87	97	1000	1000	368	1000	603
Clothing	97	20	43	27	97	83	973	167	1000	102
Coach	93	63	100	83	100	188	838	619	1000	242
Company	97	83	100	100	97	1000	1000	245	1000	784
Conference	93	53	97	90	100	95	990	437	928	92
Country	57	33	97	37	93	1000	1000	130	1000	207
EconomicSector	60	23	100	10	77	1000	1000	34	1000	138
Emotion	77	53	87	60	83	483	992	183	1000	211
Food	90	70	97	80	100	811	1000	89	1000	272
Furniture	100	0	57	57	90	55	963	215	1000	95
Hobby	77	33	77	50	90	357	936	77	1000	127
KitchenItem	73	3	88	13	100	11	900	8	960	2
Mammal	83	50	93	50	90	224	1000	154	1000	169
Movie	97	57	97	100	100	718	1000	566	1000	183
NewspaperCompany	90	60	60	97	100	179	1000	1000	1000	241
Politician	80	60	97	37	100	178	990	30	1000	101
Product	90	83	-	77	70	1000	1000	0	999	127
ProductType	73	63	27	63	50	712	1000	31	1000	159
Profession	73	53	-	57	93	916	973	0	1000	171
ProfessionalOrganization	93	63	100	77	87	104	943	58	1000	163
Reptile	95	3	90	27	100	19	912	149	1000	54
Room	64	0	33	7	100	25	913	12	643	3
Scientist	97	30	100	17	100	83	971	928	1000	130
Shape	77	7	7	7	85	43	985	28	733	26
Sport	77	13	63	83	73	283	1000	225	1000	284
SportsEquipment	20	10	57	23	23	58	902	52	1000	174
SportsLeague	100	7	80	27	86	11	901	10	1000	14
SportsTeam	90	30	87	87	87	301	903	864	944	506
Stadium	93	57	53	63	90	102	767	944	1000	343
StateOrProvince	77	63	83	93	77	202	1000	114	1000	161
Tool	40	13	93	90	97	561	1000	713	1000	59
Trait	53	40	52	47	97	234	1000	21	1000	44
University	93	97	100	90	93	1000	1000	961	1000	516
Vehicle	67	30	50	13	77	460	1000	50	1000	98
Average	78	41	78	59	90	360	960	271	976	199
Weighted average	79	42	86	59	91					

Table 2: Precision (%) and counts of promoted instances for each category using CPL, UPL, CSEAL, SEAL, and MBL.

	Precision (%)					Promoted Instances (#)				
Predicate	CPL	UPL	CSEAL	SEAL	MBL	CPL	UPL	CSEAL	SEAL	MBL
CompanyAcquiredCompany	97	77	-	-	-	93	230	0	0	0
AthletePlaysForTeam	100	93	100	76	100	9	269	4	17	96
AthletePlaysInLeague	-	78	100	57	-	0	18	14	82	0
AthletePlaysSport	100	47	100	100	100	83	258	1	1	109
CEOOfCompany	100	100	-	100	100	18	18	0	1	1
CityLocatedInCountry	93	57	100	100	100	185	787	9	577	136
CityLocatedInState	100	70	100	93	100	76	194	34	537	54
CoachCoachesInLeague	-	-	0	-	-	0	0	1	0	0
CoachCoachesTeam	100	100	-	-	100	324	668	0	0	6
CompanyIsInEconomicSector	93	97	-	-	-	583	889	0	0	0
CompanyCompetesWithCompany	100	67	-	-	-	28	123	0	0	0
CompanyHasOfficeInCity	-	63	-	100	-	0	526	0	4	0
CompanyHasOfficeInCountry	-	90	-	-	-	0	195	0	0	0
CompanyHeadquarteredInCity	50	53	100	100	-	2	532	1	2	0
LeaguePlaysGamesInStadium	-	-	-	100	-	0	0	0	177	0
CompanyProducesProduct	97	93	-	-	100	54	215	0	0	8
ProductInstanceOfProductType	73	67	-	-	-	153	484	0	0	0
SportUsesSportsEquipment	33	3	100	87	33	15	1330	5	15	6
StadiumLocatedInCity	100	20	77	70	90	7	600	200	554	56
StateHasCapitalCity	60	70	-	73	-	266	188	0	495	0
StateLocatedInCountry	97	40	100	97	100	194	1299	46	653	61
TeamHasHomeStadium	100	87	100	100	100	97	208	179	106	92
TeamPlaysAgainstTeam	100	80	-	-	-	238	2088	0	0	0
TeamHasHomeCity	-	57	-	93	100	0	680	0	29	11
TeamPlaysInLeague	100	67	100	100	100	7	255	104	749	23
TeamPlaysSport	-	70	100	100	100	0	177	30	30	37
TeamWonAwardTrophyTournament	90	70	-	-	-	128	262	0	0	0
Average	89	69	91	91	95	95	463	23	149	26
Weighted Average	91	61	92	90	99					

Table 3: Precision (%) and counts of promoted instances for each relation using CPL, UPL, CSEAL, SEAL, and MBL.

5.1.6 *Supplementary Online Materials*

Several different types of materials from our evaluation are posted online at `http://rtw.ml.cmu.edu/wsdm10_online`:

- Seeds for all predicates.
- All instances promoted by MBL, CPL, UPL, CSEAL, and SEAL.
- All textual patterns promoted by pattern learning in the MBL, CPL, and UPL experiments.
- Browseable knowledge bases in XML format of all promoted instances and candidate instances from the runs of MBL, CPL, and CSEAL, 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.

5.2 Mutual Exclusion and Type Checking

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: The algorithm as described in Section 4.1.
- UPL: This method is an "uncoupled" version of CPL; it does not couple predicates using mutual-exclusion constraints or type checking. Relation instance arguments are not filtered using their categories and candidate instances and patterns are not filtered out based on violations of mutual exclusion. The common/proper noun specifications of arguments *are* used to filter out implausible instances.

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

- CSEAL: The algorithm as described in Section 4.2.
- SEAL: This method uses the implementation of SEAL provided by the authors of SEAL. As in the UPL method, it does not couple the learning of predicates using mutual-exclusion constraints or type checking.

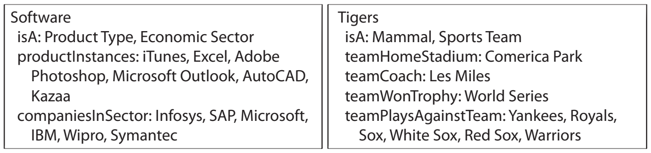
5.2.1 *Results*

Table 2 gives estimates of the precision of promoted instances for each category for each algorithm, 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. Table 3 gives this information for each relation, as well. Across all categories and relations, CPL has higher average precision than UPL, and CSEAL has higher average precision than SEAL. These results suggest that coupling using type checking and mutual exclusion significantly 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

Comparison	All Promotions Wins	All Promotions p-value	Minimum Recall Wins	Minimum Recall p-value
CPL vs. UPL	55 vs. 12	1.03e-07	37 vs. 8	1.54e-05
CSEAL vs. SEAL	36 vs. 15	0.00460	17 vs. 12	0.458
MBL vs. CPL	34 vs. 18	0.0365	28 vs. 6	0.000195
MBL vs. CSEAL	31 vs. 14	0.0161	18 vs. 6	0.0227

Table 4: Various pairs of methods 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 except for CSEAL vs. SEAL at minimum recall.



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 2: Examples of extracted facts. The generalizations 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.

hypothesis test. The test statistic needed to compare, for example, 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.

Table 4 compares CPL vs. UPL and CSEAL vs. SEAL for the precision of all promoted instances for each predicate, as well as the "minimum recall" sample discussed above. CPL performs statistically significantly better than UPL for both methods of sampling. CSEAL is significantly better than SEAL with respect to the precision of all promotions, but is not significantly better when thresholding recall to the minimum recall for each predicate. These results confirm that coupling yields significantly higher accuracies across all predicates than using independent, uncoupled learning. The results for CSEAL vs. SEAL suggest that coupling prevents CSEAL from promoting some incorrect instances but with some loss in recall.

Figure 2 gives some examples of the type of information extracted in our experiments. 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.

5.3 Multiple Extraction Techniques

Tables 2 and 3 also 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 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. Table 4 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 precision of all promoted instances as well as the precision of promoted instances at the minimum recall of either method. This suggests that coupling CPL and CSEAL with a multi-view coupling constraint that assumes independent errors yields more accurate learning than either method used alone.

5.4 Discussion

The results presented here demonstrate that adding more coupling improves the accuracy of our learned extractors.

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, Country performs poorly, drifting into a more general Location category.

The categories for which the coupled algorithms still have 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.

The coupled algorithms generally had high accuracies for relations, but suffered from sparsity. SportUsesSportsEquipment 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 in our current work. We plan to investigate learning to infer such relations.

6. CONCLUSION

We have presented methods of coupling the semi-supervised learning of category and relation instance extractors 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.

Acknowledgments

This work is supported in part by DARPA, Google, a Yahoo! Fellowship to Andrew Carlson, and the Brazilian research agencies CNPq and CAPES. We also gratefully acknowledge Jamie Callan for making available his collection of web pages, Yahoo! for use of their M45 computing cluster, and the anonymous reviewers for their comments.

7. REFERENCES

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proc. of JCDL*, 2000.

[2] Maria-Florina Balcan and Avrim Blum. A PAC-style model for learning from labeled and unlabeled data. In *Proc. of COLT*, 2004.

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

[4] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, 1998.

[5] Sergey Brin. Extracting patterns and relations from the world wide web. In *Proc. of WebDB Workshop at 6th International Conference on Extending Database Technology*, 1998.

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

[7] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

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

[9] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proc. of EMNLP*, 1999.

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

[11] Hal Daumé. Cross-task knowledge-constrained self training. In *Proc. of EMNLP*, 2008.

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

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

[14] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*, 1992.

[15] Qiuhua Liu, Xuejun Liao, Hui Li, Jason Stack, and Lawrence Carin. Semi-supervised multitask learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1074–1086, 2009.

[16] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proc. of NAACL*, 2006.

[17] Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: fact extraction in the fast lane. In *Proc. of ACL*, 2006.

[18] Marco Pennacchiotti and Patrick Pantel. Entity extraction via ensemble semantics. In *Proc. of EMNLP*, 2009.

[19] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. of AAAI*, 1999.

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

[21] 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 *Proc. of EMNLP*, 2008.

[22] 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 *Proc. of EMNLP*, 2008.

[23] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Proc. of NIPS*, 1996.

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

[25] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *Proc. of ICDM*, 2008.

[26] Richard C. Wang and William W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proc. of EMNLP*, 2009.

[27] Roman Yangarber. Counter-training in discovery of semantic patterns. In *Proc. of ACL*, 2003.

[28] Dmitry Zelenko, Chinatsu Aone, Anthony Richardella, Jaz K, Thomas Hofmann, Tomaso Poggio, and John Shawe-Taylor. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3, 2003.