Coupling Semi-Supervised Learning of Categories and Relations

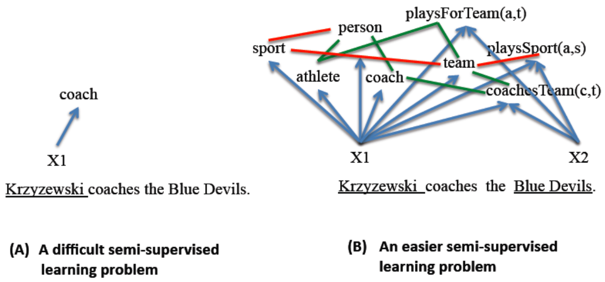
Andrew Carlson[1], **Justin Betteridge**[1], **Estevam R. Hruschka Jr.**[1,2] and **Tom M. Mitchell**[1]
[1]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
[2]Federal University of Sao Carlos
Sao Carlos, SP - Brazil

Abstract

We consider semi-supervised learning of information extraction methods, especially for extracting instances of noun categories (e.g., 'athlete,' 'team') and relations (e.g., 'playsForTeam(athlete,team)'). Semi-supervised approaches using a small number of labeled examples together with many unlabeled examples are often unreliable as they frequently produce an internally consistent, but nevertheless incorrect set of extractions. We propose that this problem can be overcome by simultaneously learning classifiers for many different categories and relations in the presence of an ontology defining constraints that couple the training of these classifiers. Experimental results show that simultaneously learning a coupled collection of classifiers for 30 categories and relations results in much more accurate extractions than training classifiers individually.

1 Introduction

A great wealth of knowledge is expressed on the web in natural language. Translating this into a structured knowledge base containing facts about entities (e.g., 'Disney') and relations between those entities (e.g. CompanyIndustry('Disney', 'entertainment')) would be of great use to many applications. Although fully supervised methods for learning to extract such facts from text work well, the cost of collecting many labeled examples of each type of knowledge to be extracted is impractical. Researchers have also explored semi-supervised learning methods that rely primarily on unlabeled data, but these approaches tend to suffer from the fact that they face an under-constrained learning task, resulting in extractions that are often inaccurate.



(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 inter-related categories and relations (B), compared with the simpler but much more difficult task of learning a single information extractor (A).

We present an approach to semi-supervised learning that yields more accurate results by coupling the training of many information extractors. The intuition behind our approach (summarized in Figure 1) is that semi-supervised training of a single type of extractor such as 'coach' is much more difficult than simultaneously training many extractors that cover a variety of inter-related entity and relation types. In particular, prior knowledge about the relationships between these different entities and relations (e.g., that 'coach(x)' implies 'person(x)' and 'not sport(x)') allows unlabeled data to become a much more useful constraint during training.

Although previous work has coupled the learning of multiple categories, or used static category recognizers to check arguments for learned relation extractors, our work is the first we know of to couple the simultaneous semi-supervised training of multiple categories and relations. Our experiments show that this coupling results in more accurate extractions. Based on our results reported here, we hypothesize that significant accuracy improvements in information extraction will be possible by coupling the training of hundreds or thousands of extractors.

2 Problem Statement

It will be helpful to first explain our use of common terms. An *ontology* is a collection of unary and binary predicates, also called *categories* and *relations*, respectively.[1] An *instance* of a category, or a *category instance*, is a noun phrase; an instance of a relation, or a *relation instance*, is a pair of noun phrases. Instances can be *positive* or *negative* with respect to a specific predicate, meaning that the predicate holds or does not hold for that particular instance. A *promoted instance* is an instance which our algorithm believes to be a positive instance of some predicate. Also associated with both categories and relations are *patterns*: strings of tokens with placeholders (e.g., 'game against *arg1*' and '*arg1* , head coach of *arg2*'). A *promoted pattern* is a pattern believed to be a high-probability indicator for some predicate.

The challenge addressed by this work is to learn extractors to automatically populate the categories and relations of a specified ontology with high-confidence instances, starting from a few seed positive instances and patterns for each predicate and a large corpus of sentences annotated with part-of-speech (POS) tags. We focus on extracting facts that are stated multiple times in the corpus, 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.

3 Related Work

Work on multitask learning has demonstrated that supervised learning of multiple "related" functions together can yield higher accuracy than learning the functions separately (Thrun, 1996; Caruana, 1997). Semi-supervised multitask learning has been shown to increase accuracy when tasks are related, allowing one to use a prior that encourages similar parameters (Liu et al., 2008). Our work also involves semi-supervised training of multiple coupled functions, but differs in that we assume explicit prior knowledge of the precise way in which our multiple functions are related (e.g., that the values of the functions applied to the same input are mutually exclusive, or that one implies the other).

In this paper, we focus on a 'bootstrapping' method for semi-supervised learning. Bootstrapping approaches start with a small number of labeled 'seed' examples, use those seed examples to train an initial model, then use this model to label some of the unlabeled data. The model is then retrained, using the original seed examples plus the 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; Ravichandran and Hovy, 2002; Pasca et al., 2006). However, after many iterations, they usually suffer from semantic drift, where errors in labeling accumulate and the learned concept 'drifts' from what was intended (Curran et al., 2007). Coupling the learning of predicates by using positive examples of one predicate as negative examples for others has been shown to help limit this drift (Riloff and Jones, 1999; Yangarber, 2003). Additionally, ensuring that relation arguments are of certain, expected types can help mitigate the promotion of incorrect instances (Paşca et al., 2006; Rosenfeld and Feldman, 2007). Our work builds on these ideas to couple the simultaneous bootstrapped training of multiple categories and multiple relations.

Our approach to information extraction is based on using high precision contextual patterns (e.g., 'is mayor of *arg1*' suggests that *arg1* 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).

[1] We do not consider predicates of higher arity in this work.

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), but emerge instead from analysis of the data. In contrast, our approach relies strongly on knowledge in the ontology about the predicates to be learned, and relationships among them, in order to achieve high accuracy.

Chang et al. (2007) present a framework for learning that optimizes the data likelihood plus constraint-based penalty terms than capture prior knowledge, and demonstrate it with semi-supervised learning of segmentation models. Constraints that capture domain knowledge guide bootstrap learning of a structured model by penalizing or disallowing violations of those constraints. While similar in spirit, our work differs in that we consider learning many models, rather than one structured model, and that we are consider a much larger scale application in a different domain.

4 Approach

4.1 Coupling of Predicates

As mentioned above, our approach hinges on the notion of coupling the learning of multiple functions in order to constrain the semi-supervised learning problem we face. Our system learns four different types of functions. For each category c:

1. $f_{c,inst} : NP(\mathcal{C}) \rightarrow [0, 1]$
2. $f_{c,patt} : Patt_C(\mathcal{C}) \rightarrow [0, 1]$

and for each relation r:

1. $f_{r,inst} : NP(\mathcal{C}) \times NP(\mathcal{C}) \rightarrow [0, 1]$
2. $f_{r,patt} : Patt_R(\mathcal{C}) \rightarrow [0, 1]$

where $\mathcal{C}$ is the input corpus, $NP(\mathcal{C})$ is the set of valid noun phrases in $\mathcal{C}$, $Patt_C(\mathcal{C})$ is the set of valid category patterns in $\mathcal{C}$, and $Patt_R(\mathcal{C})$ is the set of valid relation patterns in $\mathcal{C}$. "Valid" noun phrases, category patterns, and relation patterns are defined in Section 4.2.2.

The learning of these functions is coupled in two ways:

1. Sharing among same-arity predicates according to logical relations
2. Relation argument type-checking

These methods of coupling are made possible by prior knowledge in the input ontology, beyond the lists of categories and relations mentioned above. We provide general descriptions of these methods of coupling in the next sections, while the details are given in section 4.2.

4.1.1 Sharing among same-arity predicates

Each predicate P in the ontology has a list of other same-arity predicates with which P is *mutually exclusive*, where $mutuallyExclusive(P, P') \equiv (P(arg1) \Rightarrow \neg P'(arg1)) \wedge (P'(arg1) \Rightarrow \neg P(arg1))$, and similarly for relations. These mutually exclusive relationships are used to carry out the following simple but crucial coupling: if predicate A is mutually exclusive with predicate B, A's positive instances *and* patterns become *negative instances* and *negative patterns* for B. For example, if 'city', having an instance 'Boston' and a pattern 'mayor of *arg1*', is mutually exclusive with 'scientist', then 'Boston' and 'mayor of *arg1*' will become a negative instance and a negative pattern respectively for 'scientist.' Such negative instances and patterns provide negative evidence to constrain the bootstrapping process and forestall divergence.

Some categories are declared to be a subset of one of the other categories being populated, where $subset(P, P') \equiv P(arg1) \Rightarrow P'(arg1)$, (e.g., 'athlete' is a subset of 'person'). This prior knowledge is used to share instances and patterns of the subcategory (e.g., 'athlete') as *positive* instances and patterns for the super-category (e.g., 'person').

4.1.2 Relation argument type-checking

The last type of prior knowledge we use to couple the learning of functions is *type checking* information which couples the learning of relations with categories. For example, the arguments of the 'ceoOf' relation are declared to be of the categories 'person' and 'company'. Our approach does not promote a pair of noun phrases as an instance of a relation unless the two noun phrases are classified as belonging to the correct argument types. Additionally, when a relation instance is promoted, the arguments become promoted instances of their respective categories.

4.2 Algorithm Description

In this section, we describe our algorithm, CBL (Coupled Bootstrap Learner), in detail.

The inputs to CBL are a large corpus of POS-tagged sentences and an initial ontology with pre-

```
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 candidate instances/patterns;
      FILTER candidates;
      TRAIN instance/pattern classifiers;
      ASSESS candidates using classifiers;
      PROMOTE highest-confidence candidates;
    end
    SHARE promoted items among predicates;
  end
```

defined categories, relations, mutually exclusive relationships between same-arity predicates, subset relationships between some categories, seed instances for all predicates, and seed patterns for the categories. Categories in the input ontology also have a flag indicating whether instances must be proper nouns, common nouns, or whether they can be either (e.g., instances of ‘city’ are proper nouns).

Algorithm 1 gives a summary of the CBL algorithm. First, seed instances and patterns are shared among predicates using the available mutual exclusion, subset, and type-checking relations. Then, for an indefinite number of iterations, CBL expands the sets of promoted instances and patterns for each predicate, as detailed below.

CBL was designed 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 using a corpus containing millions of web pages (see Section 5.3 for details on the corpus).

4.2.1 Sharing

At the start of execution, seed instances and patterns are shared among predicates according to the mutual exclusion, subset, and type-checking constraints. Newly promoted instances and patterns are shared at the end of each iteration.

4.2.2 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, CBL limits 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. Candidate extraction is performed for all predicates in a single pass through the corpus using the MapReduce framework.

The candidate extraction procedure has definitions for valid instances and patterns that limit extraction to instances that look like noun phrases and patterns that are likely to be informative. Here we provide brief descriptions of those definitions.

Category Instances In the placeholder of a category pattern, CBL looks for a noun phrase. It uses part-of-speech tags to segment noun phrases, ignoring determiners. Proper nouns containing prepositions are segmented using a reimplementation of the Lex algorithm (Downey et al., 2007). Category instances are only extracted if they obey the proper/common noun specification of the category.

Category Patterns If a promoted category instance is found in a sentence, CBL extracts the preceding words as a candidate pattern if they are verbs followed by a sequence of adjectives, prepositions, or determiners (e.g., ‘being acquired by *arg1*’) or nouns and adjectives followed by a sequence of adjectives, prepositions, or determiners (e.g., ‘former CEO of *arg1*’).

CBL 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, 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 5 tokens, has a content word, has an uncapitalized word, and has at least one non-noun.

4.2.3 Candidate Filtering

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 instances.

All co-occurrence counts needed by the filtering step are obtained with an additional pass through the corpus using MapReduce. 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.

4.2.4 Candidate Assessment

Next, for each predicate 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 positive and negative patterns associated with the class. The current sets of promoted and negative instances are used as training examples for the classifier. Attributes are discretized 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 penalize extremely rare patterns by thresholding the denominator using the 25th percentile candidate pattern hit count (McDowell and Cafarella, 2006).

All of the co-occurrence counts needed for the assessment step are collected in the same MapReduce pass as those required for filtering candidates.

4.2.5 Candidate Promotion

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

5 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?

5.1 Configurations of the Algorithm

We ran our algorithm in three configurations:

- Full: The algorithm as described in Section 4.2.
- No Sharing Among Same-Arity Predicates (NS): This configuration couples predicates only using type-checking constraints. It uses 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.
- No Category/Relation coupling (NCR): This configuration couples predicates using mutual exclusion and subset constraints, but not type-checking. It uses 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. The only type-checking information used is the common/proper noun specifications of arguments for filtering out implausible instances.

5.2 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 seed

	5 iterations			10 iterations			15 iterations		
Predicate	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 . Note that we expect Full and NCR to perform similarly for categories, but for Full to outperform NCR on relations and for Full to outperform NS on both categories and relations.

instances and 5 seed patterns. The seed instances 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). Most predicates were declared as mutually exclusive with most others, except for special cases (e.g., 'hobby' and 'sport'; 'university' and 'sports team'; and 'has offices in' and 'headquartered in').

5.3 Corpus

Our text corpus 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.

5.4 Experimental Procedure

We ran each configuration for 15 iterations. To evaluate the precision of promoted instances, we sampled 30 instances from the promoted set for each predicate in each configuration after 5, 10, and 15 iterations, pooled together 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. While samples of 30 instances do not produce tight confidence intervals around individual estimates, they are sufficient for testing for the effects in which we are interested.

5.5 Results

Table 1 shows the precision of each of the three algorithm configurations for each category and relation 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.

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).[2] 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.

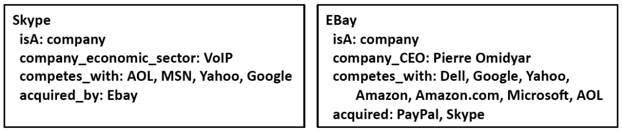
The worst-performing categories after 15 iterations of Full are 'country,' 'economic sector,' and 'hobby.' The Full configuration of CBL promoted 1637 instances for 'country,' far more than the number of correct answers. Many of these are general geographic regions like 'Bayfield Peninsula' and 'Baltic Republics.' In the 'hobby' case, promoting patterns like 'the types of *arg1*' 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.' 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 cate-

[2] 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.

	Categories		Relations	
Configuration	Instances	Prec.	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.



Skype
isA: company
company_economic_sector: VoIP
competes_with: AOL, MSN, Yahoo, Google
acquired_by: Ebay

EBay
isA: company
company_CEO: Pierre Omidyar
competes_with: Dell, Google, Yahoo, Amazon, Amazon.com, Microsoft, AOL
acquired: PayPal, Skype

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.

gories 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 and relation classifiers. Note the initial seed examples provided to CBL did not include information about either company or any of these relation instances.[3]

5.6 Comparison to an Existing Database

To estimate the capacity of our algorithm to contribute additional facts to publicly available semantic resources, we compared the complete lists of instances promoted during the Full 15 iteration run for certain categories to corresponding lists in the Freebase database (Metaweb Technologies, 2009). Excluding the categories that did not have a directly corresponding Freebase list, we computed for each category: $Precision \times |CBLInstances| - |Matches|$, where $Precision$ is the estimated precision from our random sample of 30 instances, $|CBLInstances|$ is the total number of instances promoted for that category, and $|Matches|$ is the number of promoted instances that had an exact match in Freebase. While exact matches may underestimate the number of matches, it should be noted that rather than make definitive claims, our intent here is simply to give rough estimates, which are shown in Table 3. These approximate numbers indicate a potential to use CBL to extend existing semantic resources like Freebase.

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

Table 3: Estimated numbers of "new instances" (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.

6 Conclusion

We have presented a method of coupling the semi-supervised learning of categories and relations and demonstrated empirically that the coupling forestalls the problem of semantic drift associated with bootstrap learning methods. We suspect that learning additional predicates simultaneously will yield even more accurate learning. An approximate comparison with an existing repository of semantic knowledge, Freebase, suggests that our methods can contribute new facts to existing resources.

Acknowledgments

This work is supported in part by DARPA, Google, a Yahoo! Fellowship to Andrew Carlson, and the Brazilian research agency CNPq. 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.

[3] See `http://rtw.ml.cmu.edu/sslnlp09` for results from a full run of the system.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *JCDL*.

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

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

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

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

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

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *EMNLP*.

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

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

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

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

Usama M. Fayyad and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *UAI*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *NAACL*.

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

Metaweb Technologies. 2009. Freebase data dumps. http://download.freebase.com/datadumps/.

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

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *AAAI*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *ACL*.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*.

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

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL*.

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

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In *NIPS*.

Peter D. Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *EMCL*.

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

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *ACL*.