A Comparison Of Efficacy And Assumptions Of Bootstrapping Algorithms For Training Information Extraction Systems

Rayid Ghani* **and Rosie Jones**†

*Accenture Technology Labs
Chicago, IL 60601, USA
rayid.ghani@accenture.com

†School of Computer Science
Carnegie Mellon University, Pittsburgh PA 15213, USA
rosie.jones@cs.cmu.edu

Abstract

Information Extraction systems offer a way of automating the discovery of information from text documents. Research and commercial systems use considerable training data to learn dictionaries and patterns to use for extraction. Learning to extract useful information from text data using only minutes of user time means that we need to leverage unlabeled data to accompany the small amount of labeled data. Several algorithms have been proposed for bootstrapping from very few examples for several text learning tasks but no systematic effort has been made to apply all of them to information extraction tasks. In this paper we compare a bootstrapping algorithm developed for information extraction, meta-bootstrapping, with two others previously developed or evaluated for document classification; cotraining and coEM. We discuss properties of these algorithms that affect their efficacy for training information extraction systems and evaluate their performance when using scant training data for learning several information extraction tasks. We also discuss the assumptions underlying each algorithm such as that seeds supplied by a user will be present and correct in the data, that noun-phrases and their contexts contain redundant information about the distribution of classes, and that syntactic co-occurrence correlates with semantic similarity. We examine these assumptions by assessing their empirical validity across several data sets and information extraction tasks.

1. Introduction

Information Extraction systems offer a way of automating the discovery of information from text documents. Both research and commercial systems for information extraction need large amounts of labeled training data to learn dictionaries and extraction patterns. Collecting these labeled examples can be very expensive, thus emphasizing the need for algorithms that can provide accurate classifications with only a a few labeled examples. One way to reduce the amount of labeled data required is to develop algorithms that can learn effectively from a small number of labeled examples augmented with a large number of unlabeled examples.

Several algorithms have been proposed for bootstrapping from very few examples for several text learning tasks. Using Expectation Maximization to estimate maximum a posteriori parameters of a generative model for text classification (Nigam et al., 2000), using a generative model built from unlabeled data to perform discriminative classification (Jaakkola and Haussler, 1999), and using transductive inference for support vector machines to optimize performance on a specific test set (Joachims, 1999) are some examples that have shown that unlabeled data can significantly improve classification performance, especially with sparse labeled training data. For information extraction, Yangarber et al. used seed information extraction template patterns to find target sentences from unlabeled documents, then assumed strongly correlated patterns are also relevant, for learning new templates. They used an unlabeled corpus of 5,000 to 10,000 documents, and suggest extending the size of the corpus used, as many initial patterns are very infrequently occurring (Yangarber et al., 2000a; Yangarber et al., 2000b).

A related set of research uses labeled and unlabeled data in problem domains where the features naturally divide into two disjoint sets. Blum and Mitchell (Blum and Mitchell, 1998) presented an algorithm for classifying web pages that builds two classifiers: one over the words that appear on the page, and another over the words appearing in hyperlinks pointing to that page. Datasets whose features naturally partition into two sets, and algorithms that use this division, fall into the co-training setting (Blum and Mitchell, 1998). Meta-Bootstrapping (Riloff and Jones, 1999) is an approach to learning dictionaries for information extraction starting only from a handful of phrases which are examples of the target class. It makes use of the fact that noun-phrases and the partial-sentences they are embedded in can be used as two complementary sources of information about semantic classes. Similar methods have been used for named entity classification (Collins and Singer, 1999).

Although a lot of effort has been devoted to developing bootstrapping algorithms for text learning tasks, there has been very little work in systematically applying these algorithms for information extraction and evaluating them on a common set of documents. All of the previously mentioned techniques have been tested on different types of problems, with different sets of documents, under different experimental conditions, thus making it difficult to objectively evaluate the applicability and effectiveness of these algorithms. In this paper, we first describe a range of bootstrapping approaches that fall into the cotraining setting and lay out the underlying assumptions for each. We then experimentally compare the performance of each algorithm on a common set of information extraction tasks and documents and relate it to the degree to which the assumptions are satisfied in the data sets and semantic learning tasks.

2. The Information Extraction Task

The information extraction tasks we tackle in this paper involve extracting noun phrases that fall into the following three semantic classes: organizations, people and locations. It is important to note that although named entity recognizers are usually used to extract these classes, the distinction we make in this paper is to extract all noun phrases (including "construction company", "jail warden", and "far-flung ports") instead of restricting our task to only proper nouns (which is the case in standard named entity recognizers). Because our focus is extraction of general semantic classes, we have not used many of the features common in English-language named entity recognition, including ones based on sequences of charactes in upper case, and matches to dictionaries, though adding these could improve the accuracy for these classes. This is important to note since that makes it likely that our results will translate to other semantic classes which are not found in online lists or written in capital letters.

The techniques we compare here are similar to those that have been used for semantic lexicon induction (eg (Riloff and Jones, 1999)). However, we believe that the noun-phrases we extract should be taken "in context". Thus, terms we generally consider unambiguous, such as place-names or dictionary terms, can now have different meanings depending on the context that they occur in. For example, the word "Phoenix" usually refers to a location, as in the following sentence:

> *A scenic drive from Phoenix lies a place of legendary beauty.*

but can also refer to the "Phoenix Land Company", as in this sentence:

> *Phoenix seeks to divest non-strategic properties if alternate uses cannot de monstrate sustainable 20% returns on capital investment.*

We can group these types of occurences in three broad categories:

General Polysemy: many words have multiple meanings. For example, "company" can refer to a commercial entity or to companionship.

General Terms: many words have a broad meaning that can refer to entities of various types. For example, "customer" can refer to a person or a company.

Proper Name Ambiguity: proper names can be associated with entities of different types. For example, "John Hancock" can refer to a person or a company, sicne companies are often named after people.

In general, we belive that the context determines whether the meaning of the word can be further determined and that we can correctly classify the noun phrase into the semantic class by examining the immediate context, in addition to the words in the noun phrase. Therefore we approach this problem as an information extraction task, where the goal is to extract and label noun phrase instances that correspond to semantic categories of interest.

3. Data Set and Representation

As our data set, we used 4392 corporate web pages collected for the WebKB project (Craven et al., 1998) of which 4160 were used for training and 232 were set aside as a test set. We preprocessed the web pages by removing HTML tags and adding periods to the end of sentences when necessary.[1] We then parsed the web pages using a shallow parser.

We marked up the held out test data by labeling each noun phrase as one or more of (NP) instance as an organization, person, location, or none. We addressed each task as a binary classification task. Each *noun phrase context* consists of two items: (1) the noun phrase itself, and (2) and the context (an extraction pattern). We used the AutoSlog (Riloff, 1996) system to generate extraction patterns.

By using both the noun phrases and the contexts surrounding them, we provide two different types of features to our classifier. In many cases, the noun phrase itself will be unambiguous and clearly associated with a semantic category (e.g., "the corporation" will nearly always be an organization). In these cases, the noun phrase alone would be sufficient for correct classification. In other cases, the context itself is a dead give-away. For example, the context containing the pattern "subsidiary of <np>" nearly always extracts an organization. In those cases, the context alone is sufficient. However, we suspect that both the noun phrase and the context often play a role in determining the correct classification.

4. Bootstrapping Algorithms

In this section we give a brief overview of each of the algorithms we will be using for bootstrapping. We analyze how the properties and assumptions of each may affect accuracy.

4.1. Baseline Methods

Since our bootstrapping algorithms all use seed noun-phrases for an initial labeling of the training data, we should look at how much of their accuracy is based on the use of those seeds, and how much is derived from bootstrapping using those seeds. To this end, we implemented two baselines which use *only* the seeds, or noun-phrases containing the seeds, but use no bootstrapping.

4.1.1. Extraction Using Seeds Only

All the algorithms we describe use seeds as their source of information about the target class. A useful way of assessing what we gain by using a bootstrapping algorithm is to use the seeds as our sole model of information about the target class. The seeds we use for bootstrapping all algorithms are shown in Table 1.

[1] Web pages pose a problem for parsers because separate lines do not always end with a period (e.g., list items and headers). We used several heuristics to insert periods whenever an independent line or phrase was suspected.

The algorithm for seed extraction is: any noun-phrase in the test set exactly matching a word on the seed list is assigned a score of 1. All other noun-phrases are assigned the prior.

4.1.2. Head Labeling Extraction

All the bootstrapping algorithms we discuss use the seeds to perform *head-labeling* to initialize the training set. The algorithm for head labeling is: any noun-phrase in the training set whose head matches a word on the seed list is assigned a score of 1. This may not lead to completely accurate initialization, if any of the seeds are ambiguous. We will discuss this in more detail in Section 5.1.

In order to evaluate the contribution of the head-labeling to overall performance of the bootstrapping, we performed experiments using the head-labeling alone as information in order to extracted from the unseen test set.

The algorithm for *head labeling extraction* is: any noun-phrase in the test set whose head matches a word on the seed list is assigned a score of 1. All other noun-phrases are assigned the prior.

4.2. Bootstrapping Methods

The bootstrapping methods we describe fall under the cotraining setting where the features naturally partition into multiple disjoint sets, any of which individually is sufficient to learn the task. The separation into feature sets we use for the experiments in this paper is that of noun-phrases, and noun-phrase-contexts.

4.2.1. Cotraining

Cotraining (Blum and Mitchell, 1998) is a bootstrapping algorithm that was originally developed for combining labeled and unlabeled data for text classification. At a high level, it uses a feature split in the data and starting from seed examples, labels the unlabeled data and adds the most confidently labeled examples incrementally. When used in our information extraction setting, the algorithm details are as follows:

1. Initialize NPs from both positive and negative seeds
2. Use labeled NPs to score contexts
3. Select k most confident positive and negative contexts, assign them the positive and negative labels
4. Use labeled contexts to label NPs
5. Select k most confident positive and negative NPs, assign them the positive and negative labels
6. goto 2.

Note that cotraining assumes that we can accurately model the data by assigning noun-phrases and contexts to a class. When we add an example, it is either a member of the class (assigned to the positive class, with a probability of 1.0) or not (assigned to the negative class, with a probability of 0.0 of belonging to the target class). As we will see in section 5.2., many noun-phrases, and many more contexts, are inherently ambiguous. Cotraining may harm its performance through its hard (binary 0/1) class assignment.

4.2.2. CoEM

coEM was originally proposed for semi-supervised text classification by Nigam & Ghani (Nigam and Ghani, 2000) and is similar to the cotraining algorithm described above, but incorporates some features of EM. coEM uses the feature split present in the data, like co-training, but is instead of adding examples incrementally, it is iterative, like EM. It starts off using the same initialization as cotraining and creates two classifiers (one using the NPs and the other using the context) to score the unlabeled examples. Instead of assigning the scored examples positive or negative labels, coEM uses the scores associated with *all* the examples and adds *all* of them to the labeled set probabilistically (in the same way EM does for semi-supervised classification). This process iterates until the classifiers converge.

Muslea et al. (Muslea et al., 2000) extended the co-EM algorithm to incorporate active learning and showed that it has a robust behavior on a large spectrum of problems because of its ability to ask for the labels of the most ambiguous examples, which compensates for the weaknesses of the underlying semi-supervised algorithm.

In order to apply coEM to learning information extraction, we seed it with a small list of words. All noun-phrases with those words as heads are assigned to the positive class, to initialize the algorithm.

Note that coEM does not perform a hard clustering of the data, but assigns probabilities between 0 and 1 of each noun-phrase and context belonging to the target class. This may reflect well the inherent ambiguity of many terms.

4.2.3. Meta-bootstrapping

Meta-bootstrapping (Riloff and Jones, 1999) is a simple two-level bootstrapping algorithm using two features sets to label one another in alternation. It is customized for information extraction, using the feature sets *noun-phrases* and *noun-phrase-contexts* (or *caseframes*). There is no notion of negative examples or features, but only positive features and unlabeled features. The two feature sets are used asymmetrically. The noun-phrases are used as initial data and the set of positive features grows as the algorithm runs, while the noun-phrase-contexts are relearned with each outer iteration.

Heuristics are used to score the features from one set at each iteration, based on co-occurrence frequency with positive and unlabeled features, using both frequency of co-occurrence, and diversity of co-occurring features. The highest scoring features are added to the positive feature list.

Meta-bootstrapping treats the noun-phrases and their contexts asymmetrically. Once a context is labeled as positive, *all* of its co-occurring noun-phrases are assumed to be positive. However, a noun-phrase labeled as positive is part of a committee of noun-phrases voting on the next context to be selected. After a phase of bootstrapping, all contexts learned are discarded, and only the best noun-phrases are retained in the permanent dictionary. The bootstrapping is then recommenced using the expanded list of noun-phrases. Once a noun-phrase is added to the permanent dictionary, it is assumed to be representative of the positive class, with confidence of 1.0.

Class	Seeds
locations	australia, canada, china, england, france, germany, japan, mexico, switzerland, united states
organizations	inc., praxair, company, companies, dataram, halter marine group, xerox, arco, rayonier timberlands, puretec
people	customers, subscriber, people, users, shareholders, individuals, clients, leader, director, customer

Table 1: Seeds used for initialization of bootstrapping.

4.3. Active Initialization

As we saw in the discussion of head-labeling (Section 4.1.2.), using seed words for initializing training may lead to initialization that includes errors. We give measures of the rate of errors in head-labeling in Table 3. We will augment the intialization of bootstrapping by correcting those errors before bootstrapping begins, and seeing the effects on test set extraction accuracy. We call this *active initialization*, by analogy to active learning.

5. Assumptions in Bootstrapping Algorithms

The bootstrapping algorithms described in Section 4.2. have a number of assumptions in common; that initialization from seeds leads to labels which are accurate for the target class, that seeds will be present in the data, that similar syntactic distribution correlates with semantic similarity, and that noun-phrases and their contexts are redundant and unambiguous with respect to the semantic classes we are attempting to learn. We assess the validity of each of these assumptions by examining the data.

5.1. Initialization from Seeds Assumption

All the algorithms we describe use seed words as their source of information about the target class. An assumption made by all the algorithms we present is that seed words suggested by a user will be present in the data. We assess this by comparing seed density for three different tasks over two types of data, one collected specifically for the task at hand, one drawn according to a uniform random distribution over documents on the world wide web. The seeds we use for initializing bootstrapping all algorithms are shown in Table 1. We show the density of seed words in different corpora in Table 2. Note that the `people` and `organizations` classes are much more prevalent in the company data we are working with than in documents randomly obtained using Yahoo's random URL page.

Another assumption that arises from using seeds is that labeling using them accurately labels items in the target semantic class. All three algorithms initialize the unlabeled data by using the seeds to perform *head labeling*. Any noun-phrase with a seed word as its head is labeled as positive. For example, when *canada* is in the seed word list, both "eastern canada" and "marketnet inc. canada" are labeled as being positive examples. Table 3 shows the accuracy for `locations` and `people`. For `people`, some

Corpus	Class	Seed-density (/10,000)
fixed	locations	18
random		21
fixed	organizations	112
random		17
fixed	people	70
random		33

Table 2: Density of seed words per 10,000 noun-phrases in fixes corpus of company web pages, and corpus of randomly collected web pages.

Class	Accuracy
`locations`	98%
`people`	95%

Table 3: Accuracy of labeling examples automatically using seed-heads.

words were mostly unambiguous, with the exception of a few examples, "customers", which was unambigous except in prhases such as "industrial customers". The seed-word "people" also led to some training examples of questionable utility, for example "invest in people". If we learn the context "invest in", it may not help in learning to extract words for people, in the general case. Other seed-words from the `people` class proved to be very ambiguous; "leader" was most often to used to describe a company, as in the sentence "Anacomp is a world leader in digital document-management services".

We will discuss the results of correcting these errors before beginning bootstrapping in Section 6.3.

5.2. Feature Sets Redundancy Assumption

The bootstrapping algorithms we discuss all assume that there is sufficient information in each feature set (noun-phrases and contexts) to use either to label an example. However, when we look at the ambiguity of noun-phrases in the test set (Table 4) we see that 81 noun-phrases were ambiguous between two classes, and 4 were ambiguous between three classes. This means that these 85 noun-phrases (2% of the 4413 unique noun-phrases occurring in the test set) are not in fact sufficient to identify the class. This discrepancy may hurt cotraining and meta-bootstrapping more, since they assume that we can classify noun-phrases into a class with 100% accuracy.

When we examine the same information for contexts (Table 4) we see even more ambiguity. 36% of contexts are ambiguous between two or more classes.

We have another measure of the inherent ambiguity of the noun-phrases making up our target class when we measure the inter-rater(labeler) agreement on the test set. We randomly sampled 230 examples from the test collection, broken into two subsets of size 114 and 116 examples. We had four labelers label subsets with different amounts of information. The three conditions were:

- noun-phrase, local syntactic context, and full sentence (*all*)
- noun-phrase, local syntactic context (*np-context*)

Ambiguity	Class(es)	Number of NPs
	none	3574
	loc	114
1	org	451
	person	189
	loc, none	6
	org, none	31
2	person, none	25
	loc, org	6
	org, person	13
3	loc, org, none	1
	org, person, none	3

Table 4: Distribution of test NPs in classes

Ambiguity	Class(es)	Number of Pats
	none	1068
	loc	25
1	org	98
	person	59
	loc, none	51
	org, none	271
2	person, none	206
	loc, org	5
	org, person	50
3	loc, org, none	18
	org, person, none	83
4	loc, org, person, none	6

Table 5: Distribution of test patterns in classes

- noun-phrase only (*np*).

The labelers were asked to label each example with any or all of the labels `organization`, `person` and `location`. Before-hand, they each labeled 100 examples separate from those described above (in the *all* condition) and discussed ways of resolving ambiguous cases (agreeing, for example, to count "we" as both `person` and `organization` when it could be referring to the organization or the individuals in it. The distribution of conditions to labelers is shown in Figure 6.

We found that when the labelers had access to the noun-phrase, context, and the full sentence they occurred in, they agreed on the labeling 90.5% of the time. However, when one did not have the sentence (only the noun-phrase and context), agreement dropped to 88.5%. Our algorithms have only the noun-phrase and contexts to use for learning. Based on the agreement of our human labelers, we conjecture that the algorithms could do better with more information.

Labeler	Set 1 Condition	Set 2 Condition
1	NP-context	all
2	all	NP-context
3	NP	all
4	all	NP

Table 6: Conditions for inter-rate evaluation - All stands for NP, context and the entire sentence in which the NP-context pair appeared

5.3. Syntactic - Semantic Correlation Assumption

All the algorithms we address in this paper use the assumption that phrases with similar syntactic distributions have similar semantic meanings. It has been shown (Dagan et al., 1999) that syntactic cooccurrence leads to clusterings which are useful for natural language tasks. However, since we seek to extract items from a single semantic target class at a time, syntactic correlation may not be sufficient to represent our desired semantic similarity.

The mismatch between syntactic correlation and semantic similarity can be measured directly by measuring context ambiguity, as we did in Section 5.2.. Consider the context "visit <X>", which is ambiguous between all four of our classes `location`, `person`, `organization` and `none`. It occurs as a `location` in "visit our area", ambiguously between `person` and `organization` in "visit us", and as `none` in "visit our website".

Similarly, examining the ambiguous noun-phrases we see that occurring with a particular noun-phrase does not necessarily determine the semantics of a context. Three of the three-way ambiguous noun-phrases in our test set are: "group", "them" and "they". Adding "they" to the model when learning one class may cause an algorithm to add contexts which belong to a different class.

Meta-bootstrapping deals with this problem by specifically forbidding a list of 35 stop words (mainly prepositions) from being added to the dictionaries. In addition, the heuristic that a caseframe be selected by many different noun-phrases in the seed list helps prevent the addition of a single ambiguous noun-phrase to have too strong an influence on the bootstrapping. The probabilistic labeling used by coEM helps prevent problems from this ambiguity. Though we also implemented a stop-list for cotraining, its all-or-nothing labeling means that ambiguous words not on the stop list (such as "group") may have a strong influence on the bootstrapping.

6. Empirical Comparison of Bootstrapping Algorithms

After running bootstrapping with each algorithm we have two models: (1) a set of noun-phrases, with associated probabilities or scores, and (2) a set of contexts with probabilities or scores. We then use these models to extract examples of the target class from a held-out hand annotated test corpus. Since we are able to associate scores with each test example, we can sort the test results by score, and calculate precision-recall curves.

6.1. Extraction on the Test Corpus

There are several ways of using the models produced by bootstrapping to extract from the test corpus:

1. Use only the noun-phrases. This corresponds to using bootstrapping to acquire a lexicon of terms, along with probabilities or weights reflecting confidence assigned by the bootstrapping algorithm. This may have advantage over lists of terms (such as proper names) which

have no such probabilities associated with them. The probabilities allow us to sort extracted phrases and thus control whether we obtain few, highly probable members of the target class, or obtain good coverage, at the expense of accuracy. We will measure these trade-offs using precision and recall, discussed in Section 6.2..

2. Use only the contexts. In this case we discard the noun-phrases we learned during bootstrapping, and use only the contexts as extraction patterns for extracting on the test set. We extract a noun-phrase when it occurs with one of the contexts in our model, using the score assigned by that context. This may have the advantage of allowing greater generalization. Unseen words and phrases can be extracted from the test corpus, and overspecialization based on the training corpus can be avoided.

3. Use both models. To score a noun-phrase context pair in the test set, assume independence, and multiply the model noun-phrase and context scores to get a probability for the example. Noun-phrases and contexts not seen in the training corpus are given a score based on the prior probability. This has the advantage of combining all the information we acquired during training. This method is most effective for methods which assign probability-like scores (coEM and cotraining). For meta-bootstrapping, there is no natural way of combining the scores.

We experimented with these extraction methods for all three algorithms, and found that method 2, extracting using only the contexts, was by far the best for meta-bootstrapping, so all our results for meta-bootstrapping use this extraction method. CoEM and cotraining performed best with method 3, combining information from both noun-phrase and context models, so all results reported for coEM and cotraining use this extraction method.

6.2. Evaluation

We use the models to score all noun-phrase instances in the test corpus, using context-scoring for meta-bootstrapping, and noun-phrase-context scoring for coEM and cotraining, as described in Section 6.1.. Since we could select a variety of thresholds if we used our models for classifiation, depending on the target application, we use a large number of thresholds, calculating precision and recall for each. Precision is given by

$$Precision = \frac{tp_t}{tp_t + fp_t}$$

where tp_t is the number of correct examples above the threshold, and fp_t is the number of incorrect examples above the threshold. Recall is given by

$$Recall = \frac{tp_t}{tp_t + fn_t}$$

where tp_t is the number of correct examples above the threshold and fn_t is the number of correct examples below the threshold.

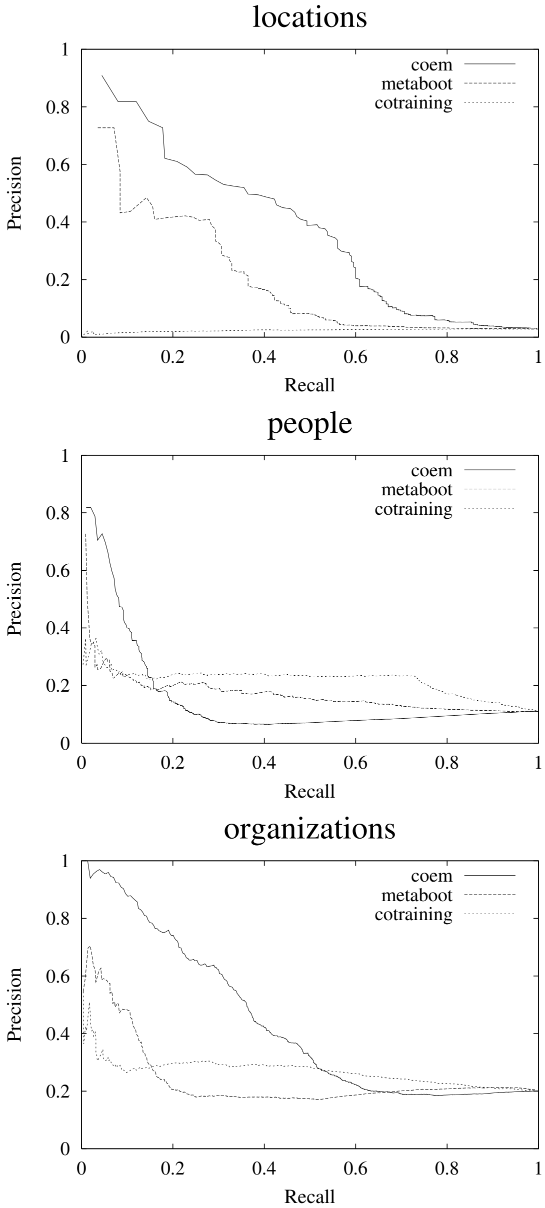

Figure 1: Comparison of bootstrapping using coEM, meta-bootstrapping and cotraining, for the classes `locations`, `people` and `organizations`.

6.3. Experimental Results

Figure 1 compares using models obtained by bootstrapping with coEM, meta-bootstrapping and cotraining, for extracting on a held out test set. CoEM performs better than meta-bootstrapping, while cotraining does very poorly.

Figure 2 shows that bootstrapping using unlabeled documents gives us significant gains over using just the seeds, or noun-phrases with the seeds as heads, for extracting from the test corpus. This difference is least marked for the class `people`, which had the most ambiguous seed words.

Figure 3 shows that only a small gain is obtained by hand-labeling all 669 examples matching the `location` seeds before commencing bootstrapping, and all 2521 examples matching the `people` class before commencing bootstrapping.

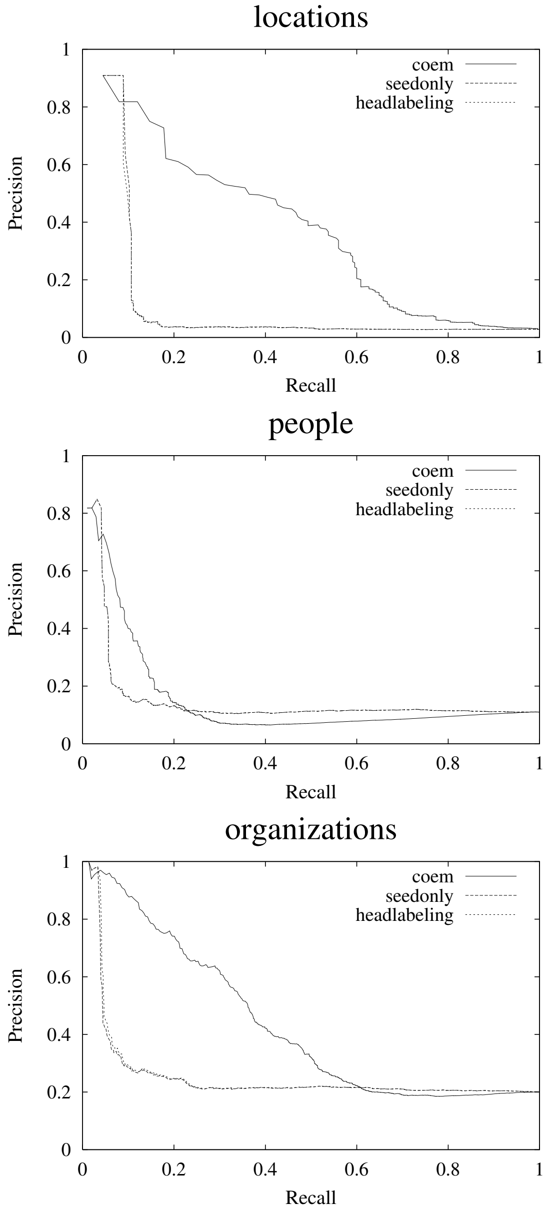

Figure 2: Comparison of the effects of using seeds alone, noun-phrases with seeds as heads (head-labeling) and models learned by bootstrapping with coEM to extract on the unseen test set. Seeds and head-labeling lead to good precision, but poor recall. Bootstrapping using coEM improves recall without loss of precision.

7. Discussion

The advantage coEM has over meta-bootstrapping and cotraining may reflect the good match between its probabilistic treatment of the data, and the inherent ambiguity of the classes. This permits an ambiguous example to be labeled with a probability that reflects its true ambiguity, rather than committing it to a class, then being overly influenced by its presence in that class. Since meta-bootstrapping repeatedly discards the contexts, ambiguity in the contexts does not hurt the algorithm as much as it hurts cotraining.

We can see from the comparison of gains from bootstrapping over using the seeds or head-labeling, that classes for which we have ambiguous seeds words, such as our

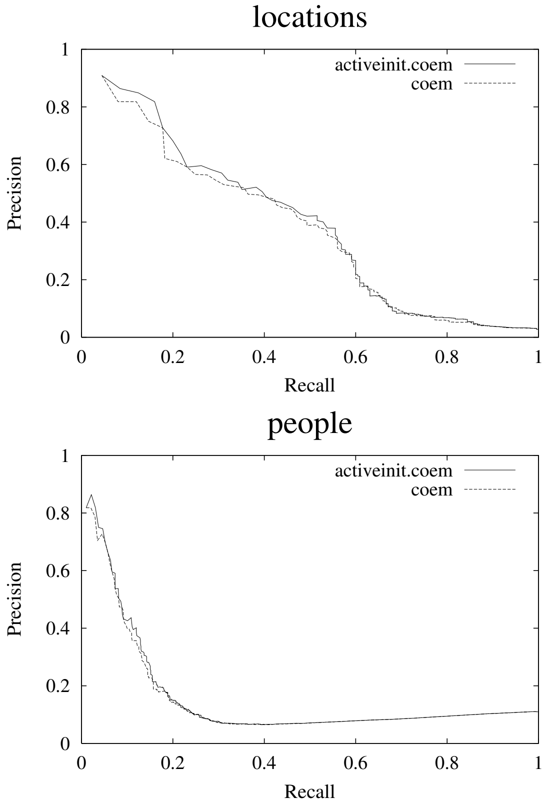

Figure 3: Comparison of the effects of hand-labeling all examples matching the seed-words before commencing bootstrapping (active initialization), against bootstrapping assuming all are correct (coem). A small gain is obtained by labeling all data input.

`people` class benefit less from bootstrapping than those with relatively unambiguous seed words. However, we still benefit from bootstrapping. This may be because the noise introduced by the ambiguous seed-words is somewhat mitigated by the presence of the less ambiguous seed words.

For `locations` and `people` we saw that correcting by hand the examples labeled using the seed words did not have a significant impact on the results. This means that for relatively unambiguous seed words, at least, hand-labeling them in context does not give us an advantage over using automatic head-labeling.

For the seed-words and datasets we used, seed density in the training corpus does not appear to be a major issue.

8. Conclusions and Future Work

We presented a range of bootstrapping algorithms for information extraction and provide experimenal results comparing cotraining, coEM and meta-bootstrapping over a common set of documents and semantic learning tasks. We also analyzed the underlying assumptions for each of the algorithms and found that performance is affected by the degree to which the assumptions are violated in the data set and the task at hand.

We also analyzed several ways of initializing the bootstrapping algorithms and found that the accuracy does not appear to hinge greatly on initialization that is 100% accurate. A greater density of seeds in the training set for a class (`organizations` and `people` had greater seed density than `locations`) does not appear to lead to greater extraction accuracy on the held out test set. Algorithms which cater to the ambiguity inherent in the feature set are more reliable for bootstrapping, whether they do that by using the feature sets asymmetrically (like meta-bootstrapping), or by allowing probabilistic labeling of examples (like coEM).

Although we have limited the scope of this paper to algorithms that utilize a feature split present in the data (co-training setting), we believe that this comparison of algorithms should be extended to settings where such a split of the features dies not exist, for examples algorithms like expectation maximization (EM) over the entire combined feature set. It would also be helpful to extend the analysis to a greater variety of semantic classes and larger sets of documents.

Acknowledgements

We thank Tom Mitchell and Ellen Riloff for numerous, extremely helpful discussions and suggestions that contributed to the work described in this paper.

9. References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1999. Similarity-based models of cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Tommi Jaakkola and David Haussler. 1999. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML '99*.

Ion Muslea, Steven Minton, and Craig A. Knoblock. 2000. Selective sampling with redundant views. In *AAAI/IAAI*, pages 621–626.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press.

E. Riloff. 1996. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. 85:101–134.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000a. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000b. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, (ANLP-NAACL 2000)*, pages 282–289.